实用第一 智慧密集 Practicability First, Intelligence Intensive

电脑编程技巧与维护

COMPUTER PROGRAMMING SKILLS & MAINTENANCE

http://www.comprg.com.cn

上 1月 2013年1月03日

每期定价:11.00元 全年定价:264.00元
《电脑编程技巧与维护》杂志社出版
刊号: ISSN 1006-4052 CN11-3411/TP
广告许可证 京海工商广字0151

国家级科技期刊 中国学术期刊综合评价数据库统计源期刊 中国核心期刊（遴选）数据库收录期刊

第84页.《全新的用户体验，瑞星杀毒软件V16成功上线！》

ISSN 1006-4052

2013 年第 01 期
1 月(上)

电脑编程技巧与维护

总第 271 期　　1994 年 7 月创刊　　(半月刊)

刊　　号：ISSN 1006-4052
CN11-3411/TP

广告订可证：京海工商广字 0151 号
全 年 定 价：264 元
每 期 定 价：11 元

电脑编程技巧与维护(半月刊)

每月3日出版

实用第一 智慧密集

目次

WiGig：超高速的无线传输技术

WiGig一种新的无线传输技术，它将让你抛弃U盘，拔掉存储设备之间原有的连线。它的特点是距离短、速度快，并且还能做得更多，比如将iPhone中的视频直接发送到电视上供全家人欣赏。

Wireless Gigabit Alliance联盟（以下简称WGA联盟）负责开发和制定WiGig。这种技术可以做到最快700MB/s的传输速度，是Wi-Fi的10倍以上。想象一下，一部10G的高清电影十几秒就可以从一台手机、平板或者电脑传输到另外一台设备上。

WiGig可以看成是Wi-Fi的一个进化。IEEE 802.11标准是Wi-Fi技术的标准。此标准的最新版本IEEE 802.11ac其最高频段达到5GHz，理论上，最快传输速度可以达到130MB/s。但5GHz仍是一个很低的频段，在低频段上无论使用任何技术都很难再让速度有一个质的加速。而WiGig使用了Wi-Fi无法使用的60GHz频段，所以它的速度成了130MB/s的近乎6倍。目前，WiGig规范已经成为了IEEE 802.11ad标准草案的基础。WiGig不是一个新的标准，而是对Wi-Fi标准进行了延伸和完善。

早在2010年5月，WGA联盟宣布和Wi-Fi联盟合作，向下兼容Wi-Fi，提出了三段式Wi-Fi的概念。用户可以根据需要自由切换2.4GHz、5GHz的Wi-Fi和WiGig。

WiGig在信号的传输方式上发生了很大的变化。通常情况下，WiGig信号可以在10米的半径收发。只是，发射出的信号从Wi-Fi时的波纹式变成了近乎直线式，它带来的优点是减少了信号的浪费。但同时，一旦在这个直线之间有了阻挡物，比如房间的墙壁，WiGig信号可能变弱。WGA联盟不是没注意到这个问题。在2010年发布WiGig第一个技术标准——WiGig1.0之时，联盟就为WiGig加入了适应性束波成形（Adagtive Beam Forming，简称ABF）技术，简单来说就是在信号发射前先扫描周围环境，若接受设备之间没有障碍物则采用直线传输，有障碍物则找一些次优路线——障碍物之外的空当来反射信号。

“WiGig规范从一开始就设计用于在各种各样的设备上工作，因此决定了在我们开始定义Wi-Fi在60GHz的认证项目时，就必须选择WiGig标准。”Wi-Fi联盟首席执行官Edgar Figueroa说。不仅仅是Wi-Fi联盟，还有VESA联盟、推广USB标准的USB-IF协会等，WiGig都进行了合作。

WiGig成为下一代技术标准的悬念不大。很多生产厂商已经陆续有WiGig产品发布。

超级计算机技术创造虚拟大脑

据外媒资料介绍，加拿大研究人员利用超级计算机技术，创造了一个具备简单认知能力的虚拟大脑，该成果有望帮助人类更好地了解大脑运作。

据加拿大滑铁卢大学的研究人员介绍，这个名为“Spaun”的虚拟大脑主体是个基于超级计算机构建的数字模型，它通过一个类似摄像镜头的仪器来观察，并可指挥机械臂进行书写等动作，更重要的是，系统中还包括250万个模拟“神经元”，它们能通过变化的电压来模拟脑电波。

“Spaun”可执行多项简单的认知任务，对别人提出的问题以及通过虚拟“眼睛”观察到的事物作出回应。例如，研究人员向“Spaun”展示数字“2”的不同写法图片后，它可以根据写法的不同重新画出这个数字。它还有不错的记忆力，可依次将之前看到的一连串数字写出来。

研究人员说，“Spaun”是首个能模拟大脑利用不同区间沟通来展示复杂行为的模型，但目前它在功能性上还远远无法与真正的大脑相比。

此前也有不少利用超级计算机模拟大脑功能的项目，但滑铁卢大学的研究人员说，“Spaun”与它们的最大不同是，此前的项目仅模拟大脑的功能形式，而“Spaun”则能展示这些功能如何作用于各种行为。

苹果研究更实用的区域无线充电技术

根据美国专利商标局的专利文件透露，苹果目前正在致力于研究一项更加实用、有效地使用无线充电的技术。据悉这项技术依靠的是“近场磁共振技术“，将能为距离1米以内的低电量设备无线供电。

苹果在专利文件中表示，无线电力传输以往仅在有数的几款应用中实现，而且这项技术要求电源与接收器距离要靠得非常近。对那些仅需小电量的设备来说这个方法尚且可行，但是对那些动辄几百瓦需要电量的设备来说这个方法就实在令人难以接受了。

苹果在专利文件中指出，实际上电力可以通过“近场”从电源方传送到接收器上，在大多数情况下，这个“近场”直径范围将达到1米。

通过采用无线充电技术，苹果就能最小化，或完全摆脱当前这种笨拙的、需要接插头的充电方法。为了能提高无线电力传输的效率，苹果的系统甚至还比“近场”更进一步。将会允许许多外设也能利用“近场”进行无线充电，比如说在“近场”范围内的鼠标、键盘等。

2012年优秀作者名单

(拼音为序)

江　洪	天津	穆宣社	天津
李　斌	上海	童小明	江西 新余
李朝中	北京	徐怀平	杭州
刘　班	湖北 荆州	杨　东	合肥
马创新	南京	张中红	河南 洛阳

基于 Django 快速开发社会化程序代码共享系统（三）

——相关功能页面和好友管理功能的实现

刘 班

摘 要：以 Django1.2.5 为基础开发了一个典型的社会化程序代码共享系统。详细讲解了该系统中相关功能页面和好友管理功能的实现。

关键词：Python 语言；Django 框架；社会化；程序代码共享系统

上期详细讲解了本系统中程序代码相关操作的实现过程，下面讲解本系统中包含的用户页面、程序代码页面以及好友管理功能（发送添加好友请求、添加好友请求列表页面、接受添加好友请求、删除添加好友请求、用户好友列表页面、删除用户好友）的实现过程。

1 用户页面

用户页面中按更新时间先后顺序以分页列表的形式显示了用户个人发布的所有程序代码，如图 1 所示。用户可以通过单击导航条或者程序代码列表中的用户名链接进入到自己或他人的个人页面中。

Django社会化程序代码共享系统

您发布的程序代码

标题	编程语言	发布时间
windows下模式删除	Python	2011-11-13 *18 小时，34 分钟前更新*
正态分布概率计算	Python	2012-03-03 *18 小时，35 分钟前更新*
用python3抓取中文网页	Python	2012-05-21 *18 小时，35 分钟前更新*
统计代码行数	Python	2012-02-26 *18 小时，36 分钟前更新*
Python随机生成中文验证码	Python	2012-06-22 *18 小时，36 分钟前更新*

图 1 用户页面

用户页面的显示由在 views.py 文件中定义的视图函数 user_page 负责实现，该视图函数的定义如下所示：

```
""定义视图函数 user_page,其参数 username 用于接收 url 请求地址中包含的用户名信息。""
def user_page(request, username):
  user = get_object_or_404(User, username=username)
 is_friend = True
 try:
  Friendship.objects.get(
   from_friend__username=request.user.username,
   to_friend=user
   )
except Friendship.DoesNotExist:
  is_friend = False
 ""如果要访问的是当前登录用户的个人页面。""
if request.user == user:
  is_friend = True
  snippets = user.snippet_set.order_by('-updated_date')
  page_title = '您发布的程序代码'
  show_rss = False
 ""如果要访问的是其他用户的个人页面。""
else:
  snippets = user.snippet_set.order_by('-updated_date')
  page_title = u'%s 发布的程序代码' % username
  show_rss = True
 return list_detail.object_list(
  request,
  queryset=snippets,
  paginate_by=ITEMS_PER_PAGE,
  template_name='user_page.html',
  template_object_name='snippet',
  extra_context={
      'username': username,
   'is_friend': is_friend,
   'page_title': page_title,
   'show_user': False,
   'show_rss': show_rss
   }
   )
```

视图函数 user_page 在执行过程中要通过模板文件 user_page.html 来生成用户页面的 HTML 代码，该模板文件的内容如下所示：

```
<! --此文件要以 utf-8 编码保存-->
{% extends "base.html" %}
{% block title %}{{page_title}}{{block.super}}{% endblock %}
{% block head %}{{block.super}}{% endblock %}
{% block content_title %}
 {{page_title}}
 {% if show_rss %}
  <a href="/feeds/user/{{username}}/rss.xml">
```

```
    <img src="/site_media/images/feed_icon.png" alt=" 订阅
RSS" border=0></img>
   </a>
  {% endif %}
{% endblock %}
{% block content %}
  {% if not is_friend %}
    <a  href = " {%  url  friend_request  % }?username =
{{username}}">请求加为好友</a>
  {% else %}
   {% ifnotequal user.username username %}
     <a  href = " {%  url  delete_friend  % }?fn ={{username}}"
onclick="return delFriendConfirm()">将该用户从您的好友列表
中删除</a>
   {% endifnotequal %}
  {% endif %}
  {% if snippet_list %}
    {% include "snippet_list.html" %}
    {% include "paginator.html" %}
  {% else %}
    <p>{% ifnotequal user.username username %}该用户{%
else %}您{% endifnotequal %}目前还没有发布程序代码!</p>
  {% endif %}
{% endblock %}
```

为了能将用户名链接与视图函数 user_page 关联起来，还需要在 urls.py 文件中添加一个视图函数 user_page 的命名 url 访问入口，如下所示：

```
url  (r'^users/(?P <username >\w + )/$', user_page, name =
'user_page'),
```

这样，用户就可以通过 url 地址 http://localhost:8000/users/<用户名>/来访问<用户名>指定用户的个人页面了。

2 程序代码

程序代码页面中包含了很重要的程序代码显示功能，如图 2 所示。

图 2 程序代码页面

程序代码不同于一般的文本信息，其显示需要使用语法高亮的特殊表现形式。程序代码语法高亮显示功能既可以在服务器端实现也可以在客户端实现，但在服务器端对程序代码进行语法高亮处理会明显增加服务器的负担。因此，本系统借助 JQuery 的一个语法高亮插件 Snippet 在客户端浏览器中对程序代码的语法高亮显示直接进行处理。在这样的设计思想指导下，实现程序代码页面显示功能的视图函数 snippet_page 变得非常简单，该视图函数定义如下所示：

```
"""定义视图函数 snippet_page,其参数 snippet_id 用于接收 url
请求地址中包含的将要显示程序代码的 id 信息。"""
def snippet_page(request, snippet_id):
  snippet = get_object_or_404(Snippet, pk=snippet_id)
  try:
    Bookmark.objects.get(snippet=snippet, user_username=
request.user.username)
    bookmarked = True
  except Bookmark.DoesNotExist:
    bookmarked = False
  """调用函数 object_detail 显示变量 snippet 中保存的 Snippet
数据模型对象的详细内容。"""
 return list_detail.object_detail(
   request,
    """参数 queryset 用于指定将要显示的 Snippet 数据模型对
象所在的 Snippet 数据模型对象查询集。"""
   queryset=Snippet.objects.all(),
    """参数 object_id 用于指定将要显示的 Snippet 数据模型对
象的主键值。"""
   object_id=snippet_id,
   template_name='snippet_page.html',
   template_object_name='snippet'
   extra_context={
    'bookmarked': bookmarked
   }
   )
```

视图函数 snippet_page 利用模板文件 snippet_page.html 生成程序代码页面的 HTML 代码，该模板文件的内容如下所示：

```
<! --此文件要以 utf-8 编码保存-->
{% extends "base.html" %}
{% load comments %}
{% block title %}{{snippet.title}}{{block.super}}{% endblock %}
{% block external %}
  <! --链接使用 JQuery 插件 Snippet 所必需的 js 脚本文件
jquery.snippet.min.js-->
   <script  type = "text/javascript"  src = "/site_media/js/jquery.
snippet.min.js"></script>
  <! --链接使用 JQuery 插件 Snippet 所必需的 css 层叠样式
表文件 jquery.snippet.min.css-->
  <link rel="stylesheet" type="text/css" href="/site_media/css/
jquery.snippet.min.css">
  <script type="text/javascript">
```

FOLLOW MASTER PROGRAM

```
$(document).ready(
 function()
 {
   $("select.style").change(
       function()
       {
           //按照程序代码所使用的程序设计语言对程序代
//码进行语法高亮处理
       $("pre.code").snippet("{{snippet.language.slug}}",
               {style: $ (this).val (), clipboard:"/site_media/swf/
ZeroClipboard.swf", showNum:true});
        });
      $("select.style").selectedIndex = 0;
      $("select.style").trigger("change");
    });
  </script>
{% endblock %}
{% block head %}{{block.super}}{% endblock %}
{% block content_menu %}
 <div class="menu">
    <! --如果当前正在查看程序代码的用户就是该程序代码
的发布者-->
   {% ifequal user snippet.user %}
    <a href="{% url snippet_edit_page snippet.id %}">[编辑]
</a> 
    <a href="{% url delete_snippet %}?snippet_id={{snippet.
id}}" class="del_snippet" onclick="return delSnippetConfirm()">
[删除]</a>
    <! --如果当前正在查看程序代码的用户不是该程序代码
的发布者-->
   {% else %}
    <img src="/site_media/images/inc.jpg" alt="对我有用"></
img ><a href = " {% url vote % }?snippet_id ={{snippet.id}}
&vote_type =inc" > 对 我 有 用 ({{snippet.votes_useful}}) </a >
    <img src = "/site_media/images/dec.jpg"
alt=" 对我没用"></img><a href="{% url vote %}?snippet_id=
{{snippet.id}}&vote_type =dec" > 对 我 没 用 ({{snippet.
votes_useless}})</a>  
      <! --如果当前登录用户已经收藏了保存在模板变量
snippet 中的 Snippet 数据模型对象-->
    {% if bookmarked %}
      <a href = " {% url delete_bookmark % }?snippet_id =
{{snippet.id}}" class = "del_bookmark" onclick = "return
delBookmarkConfirm()">[取消收藏]</a>
      <! --如果当前登录用户没有收藏保存在模板变量
snippet 中的 Snippet 数据模型对象-->
      {% else %}
        <a href = " {% url add_bookmark % }?snippet_id =
{{snippet.id}}">[收藏]</a>
      {% endif %}
   {% endifequal %}
 </div>
{% endblock %}
{% block content_title %}{{snippet.title}}{% endblock %}
{% block content %}
 <p>
   <b > 发 布 者 : </b > <a href = " {{snippet.user.
get_absolute_url}}" >{{snippet.user.username}} </a > 
   <b>发布时间:</b>{{snippet.pub_date|
date:"Y-m-d"}}    <b>更新时间:</
b>{{snippet.updated_date|timesince}}前    
 <b>语言:</b><a href="{% url main_page %}?lang=
{{snippet.language.slug}}" >{{snippet.language.name}} </a >
    
     <b>代码显示主题:</b>
    <! --Snippet 插件支持很多不同类型的程序代码高亮主
题,本系统仅仅列出了其中的一部分主题供用户选择,更多具体
的主题信息可以在 Snippet 插件的项目网站 http://www.
steamdev.com/snippet/中查看。-->
   <select class="style">
    <option value="acid" selected>acid</option>
    <option value="bipolar">bipolar</option>
      <option value="desert">desert</option>
      <option value="dull">dull</option>
      <option value="easter">easter</option>
      <option value="emacs">emacs</option>
      <option value="golden">golden</option>
      <option value="greenlcd">greenlcd</option>
  </select><br>
    <b>标签:</b>
    {% if snippet.tag_set.all %}
   {% for tag in snippet.tag_set.all %}
     <a href="{{tag.get_absolute_url}}">{{tag.name}}</a>
   {% endfor %}
  {% else %}
    无
  {% endif %}
</p>
<pre class="code">{{snippet.code}}</pre>
<b>描述:</b>
{% if snippet.description %}
 {{snippet.description}}
{% else %}
 无
 {% endif %}
 <h2>用户评论</h2>
 {% get_comment_list for codesharing.snippet snippet.id as
comment_list %}
 {% if comment_list %}
  {% for comment in comment_list %}
   <div class="comment">
     <b ><a  href = " {{comment.user.get_absolute_url}}" >
{{comment.user.username}}</a></b>   
     <i>发表于{{comment.submit_date|date:"Y-m-d H:i:s"}}
```

```
   {{comment.ip_address}}</i>
      {{comment.comment|escape|urlizetrunc:40|linebreaks}}
    </div>
  {% endfor %}
{% else %}
  <p>目前还没有用户对该程序代码进行过评论！</p>
{% endif %}
 <! --生成评论输入表单-->
{% render_comment_form for codesharing.snippet snippet.
id %}
{% endblock %}
```

标签 render_comment_form 需要借助本系统 comments 文件夹中的模板文件 form.html 来生成评论输入表单的 HTML 代码，该模板文件的内容如下所示：

```
<! --此文件要以utf-8 编码保存-->
{% load comments %}
{% if user.is_authenticated %}
  <form name="comment_form" class="comment" action="
{% comment_form_target %}" method="post">
    {% csrf_token %}
      {% for field in form %}
        {% if field.is_hidden %}
          {{ field }}
        {% endif %}
    {% endfor %}
      <label for="id_comment">发表评论:</label>
    <textarea name="comment" id="id_comment" rows="10"
cols="60"></textarea>
    <input type="submit" name="post" value="提交评论"/>
  </form>
{% else %}
    <p > 请 <a  href = "/accounts/login/?next ={{request.
get_full_path}}">登录</a>后,再发表评论。</p>
{% endif %}
```

视图函数 snippet_page 的命名 url 访问入口在 urls.py 文件中添加，如下所示：

```
url (r'^snippet/(?P<snippet_id>\d+)/$', snippet_page, name=
'snippet_page'),
```

这样，用户就可以通过 url 地址 http://localhost:8000/snippet/<程序代码 id>/来进入<程序代码 id>指定程序代码的页面了。

3 好友管理相关功能

3.1 发送添加好友请求

好友管理功能是社会化 Web 应用系统中包含的一个必不可少的重要功能，本系统所包含的好友管理功能主要由发送添加好友请求、浏览添加好友请求列表、接受添加好友请求、删除添加好友请求、浏览用户好友列表以及删除用户好友这 6 个子功能组成。

如果用户 A 想添加另一个用户 B 为好友，那么用户 A 必须先进入用户 B 的个人页面，单击该页面中的“请求加为好友”链接，向用户 B 发送添加好友请求。实现这一功能的视图函数是 friend_request，其定义如下所示：

```
@login_required
"""定义视图函数 friend_request。"""
def friend_request(request):
  """如果以 get 方法请求的数据中包含参数 username(该参数
的值用于指定接收添加好友请求的用户名信息)。"""
  if request.GET.has_key('username'):
    username = request.GET['username']
  recipient = get_object_or_404(User, username=username)
    """ 如果接收添加好友请求的用户就是发送该请求的用户
(即当前登录用户)。"""
  if recipient == request.user:
    msg = ' 不能向自己发送添加好友请求！'
    """如果接收添加好友请求的用户已经存在于发送该请求用
户的好友列表中了。"""
  elif recipient in  [friendship.to_friend for friendship in
request.user.friend_set.all()]:
    msg = u' 用户%s 已经是您的好友了，无需向其发送添加
好友请求！' % recipient.username
    """如果接收添加好友请求的用户已经存在于发送该请求用
户以前发送过的添加好友请求的接收请求用户列表中了。"""
  elif recipient in  [fr.recipient for fr in request.user.sent_frs.
all()]:
    msg = u' 您已经向用户%s 发送过添加好友请求，无需重
复发送！' % recipient.username
  else:
    new_fr = FriendRequest.objects.create(sender=request.
user, recipient=recipient)
    msg = u' 您的添加好友请求已经发送给用户%s 了！' %
recipient.username
    request.user.message_set.create(message=msg)
    return  HttpResponseRedirect (recipient.get_absolute_url
())
  else:
   raise Http404(' 参数错误！')
```

为了能正常触发视图函数 friend_request 的执行，还需要在 urls.py 文件中添加一个该视图函数的命名 url 访问入口，如下所示：

```
url(r'^fr/$', friend_request, name='friend_request'),
```

这样，用户就可以通过 url 地址 http://localhost:8000/fr/?username=<用户名>向<用户名>指定的用户发送添加好友请求了。

3.2 添加好友请求列表页面

用户登录本系统之后，如果有其他用户向该用户发送过添加好友请求，则在导航条中就会出现相应的提示信息链接，用户可以通过单击该链接进入添加好友请求列表页面，在该页面

中显示了添加好友请求的详细信息和处理请求的链接，如图 3 所示。

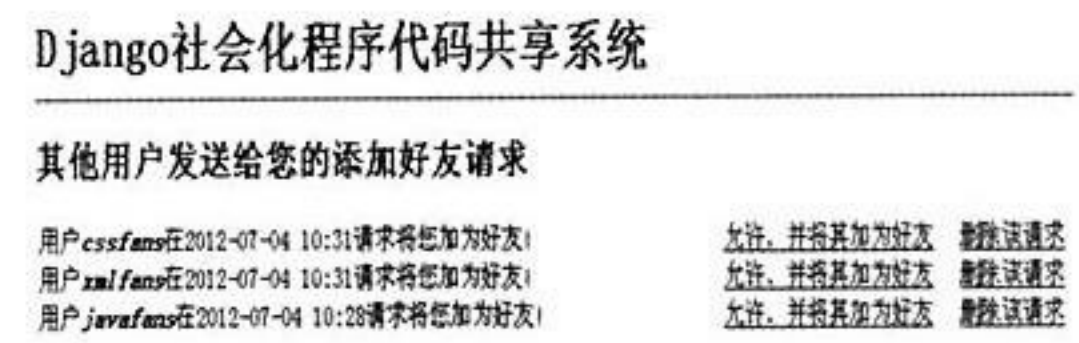


图 3　添加好友请求列表页面

实现添加好友请求列表页面显示的视图函数是fr_list_page，其定义如下所示：

```
@login_required
""定义视图函数 fr_list_page。""
def fr_list_page(request):
  return list_detail.object_list(
   request,
   queryset=request.user.received_frs.all(),
   paginate_by=ITEMS_PER_PAGE,
   template_name='fr_list_page.html',
   template_object_name='fr'
   )
```

视图函数 fr_list_page 在执行过程中要使用模板文件 fr_list_page.html 生成添加好友请求列表页面的 html 代码，该模板文件的内容如下所示：

```
<! --此文件要以 utf-8 编码保存-->
{% extends "base.html" %}
{% block title %} 其他用户发送给您的添加好友请求{{block.
super}}{% endblock %}
{% block head %}{{ block.super}}{% endblock %}
{% block content_title %} 其他用户发送给您的添加好友请求
{% endblock %}
{% block content %}
 {% if fr_list %}
  <table>
   {% for fr in fr_list %}
    <tr>
        <td  class = "friendrequest" > 用 户 <b ><i >{{fr.
sender.username}}</i></b>在 {{fr.sent_date|date:"Y-m-d H:
i"}}请求将您加为好友！ </td>
        <td>  <a href="{% url accept_fr %}?
fn={{fr.sender.username}}">允许，并将其加为好友</a></td>
        <td>  <a href="{% url delete_fr %}?
fr_id={{fr.id}}">删除该请求</a></td>
    </tr>
   {% endfor %}
  </table>
 {% include "paginator.html" %}
{% else %}
 <p>目前还没有用户向您发送过添加好友请求！ </p>
 {% endif %}
{% endblock %}
```

视图函数 fr_list_page 的命名 url 访问入口在 urls.py 文件中添加，如下所示：

```
url(r'^frs/$', fr_list_page, name='fr_list_page'),
```

3.3　接受添加好友请求

在添加好友请求列表页面中列出的每一个添加好友请求的后面都有一个“允许，并将其加为好友”的接受请求链接，用户可以通过单击该链接与发送相应请求的用户互相加为好友。实现此功能的视图函数是 accept_fr，其定义如下所示：

```
@login_required
""定义视图函数 accept_fr。""
def accept_fr(request):
  ""如果以 get 方法请求的数据中包含参数 fn(该参数的值用于指定发送了添加好友请求的用户名信息)。""
  if request.GET.has_key('fn'):
    fn = request.GET['fn']
   friend = get_object_or_404(User, username=fn)
    ""如果要添加为当前登录用户好友的用户(由其发出添加好友请求)就是当前登录用户(由其接受添加好友请求)。""
   if friend == request.user:
    msg = '不能将自己添加为好友！'
    ""如果要添加为当前登录用户好友的用户并不存在于向当前登录用户发送过添加好友请求的发送者列表中。""
   elif friend not in [fr.sender for fr in request.user.
received_frs.all()]:
    msg = u'用户%s 未向您发送过添加好友请求，或者他已经成为了您的好友！' % friend.username
    ""如果要添加为当前登录用户好友的用户既存在于被当前登录用户添加为好友的用户列表中，又存在于将当前登录用户添加为好友的用户列表中。""
   elif (friend in [friendship.to_friend for friendship in request.
user.friend_set.all()]) and (friend in [friendship.from_friend for
friendship in request.user.to_friend_set.all()]):
     msg = u'您和用户%s 之间已经建立了好友关系，无需重复建立！' % friend.username
   else:
    friendship, dummy = Friendship.objects.get_or_create(
     from_friend=friend,
     to_friend=request.user
     )
    friendship, dummy = Friendship.objects.get_or_create(
     from_friend=request.user,
     to_friend=friend
     )
    fr = get_object_or_404 (FriendRequest, sender=friend,
recipient=request.user)
    fr.delete()
    msg = u'您和用户%s 之间已经成功建立了好友关系！' %
 friend.username
```

```
    request.user.message_set.create(message=msg)
    return HttpResponseRedirect(friend.get_absolute_url())
  else:
    raise Http404('参数错误！')
```

为了能正常触发视图函数 accept_fr 的执行，还需要在 urls.py 文件中添加一个该视图函数的命名 url 访问入口，如下所示：

```
url(r'^fr/accept/$', accept_fr, name='accept_fr'),
```

这样，用户就可以通过 url 地址 http://localhost:8000/fr/accept/?fn=<用户名>来接受<用户名>指定的用户发送过来的添加好友请求，与该用户互相加为好友了。

3.4 删除添加好友请求

在添加好友请求列表页面中，用户可以通过单击“删除该请求”链接将其对应的添加好友请求删除。实现此功能的视图函数是 delete_fr，其定义如下所示：

```
@login_required
"""定义视图函数 delete_fr。"""
def delete_fr(request):
  """如果以 get 方法请求的数据中包含参数 fr_id(该参数的值用于指定将要被删除的添加好友请求的 id 信息)。"""
  if request.GET.has_key('fr_id'):
    fr_id = request.GET['fr_id']
    fr = get_object_or_404(FriendRequest, pk=fr_id)
    try:
      fr.delete()
      msg = u'已经成功删除用户%s 发送给您的添加好友请求！' % fr.sender.username
    except:
      msg = u'未能成功删除用户%s 发送给您的添加好友请求！' % fr.sender.username
    request.user.message_set.create(message=msg)
    return HttpResponseRedirect(reverse('fr_list_page'))
  else:
    raise Http404('参数错误！')
```

视图函数 delete_fr 的命名 url 访问入口在 urls.py 文件中添加，如下所示：

```
url(r'^fr/delete/$', delete_fr, name='delete_fr'),
```

这样，用户就可以通过 url 地址 http://localhost:8000/fr/delete/?fr_id=<添加好友请求 id>来删除<添加好友请求 id>指定的添加好友请求了。

3.5 用户好友列表页面

用户登录本系统之后，可以通过单击导航条中的“好友”链接进入自己的好友列表页面，在该页面中用户可以查看的内容包括：自己的好友、将自己添加为好友的用户以及好友近期发布的程序代码，如图 4 所示。

实现用户好友列表页面显示功能的视图函数是 friend_list_page，其定义如下所示：

图 4 用户好友列表页面

```
@login_required
"""定义视图函数 friend_list_page。"""
def friend_list_page(request):
  to_friends = [friendship.to_friend for friendship in request.user.friend_set.all()]
  from_friends = [friendship.from_friend for friendship in request.user.to_friend_set.all()]
  friend_snippets = Snippet.objects.filter(user__in = to_friends).order_by('-pub_date')
  return list_detail.object_list(
    request,
    queryset=friend_snippets[:30],
    template_name='friend_list_page.html',
    template_object_name='snippet',
    extra_context={
      'to_friends': to_friends,
      'from_friends': from_friends,
      'show_user': True
    }
  )
```

视图函数 friend_list_page 在执行过程中要利用模板文件 friend_list_page.html 生成用户好友列表页面的 html 代码，该模板文件的内容如下所示：

```
<!--此文件要以 utf-8 编码保存-->
{% extends "base.html" %}
{% block title %}您的好友{{block.super}}{% endblock %}
{% block head %}{{block.super}}{% endblock %}
{% block content_title %}您的好友{% endblock %}
{% block content %}
  <h3>好友列表</h3>
  {% if to_friends %}
    <ul class="friends">
      {% for to_friend in to_friends %}
        <li class = "friend" ><a href = " {{to_friend.get_absolute_url}}">{{to_friend.username}}</a></li>
      {% endfor %}
    </ul><br>
  {% else %}
```

(下转第 13 页)

任意窗口置顶器的设计与实现

李 斌

摘　要： 通过使用 Win32 API 函数 EnumWindows 和 SetWindowPos，达到枚举系统所有窗口，并使某个窗口置顶，以此设计一个任意窗口置顶器。

关键词： EnumWindows 函数；SetWindowPos 函数；窗口置顶

1　背景知识

Windows 操作系统带有一个“任务管理器”程序，利用此程序可以查看系统中所有正在运行的进程等信息，读者可能注意到，此程序默认总是“置顶”，这样可方便用户查看信息。所谓“置顶”是指该程序的主窗口位于所有其他窗口层次之上，这样就做到了凡是该程序窗口覆盖的位置，其他窗口都无法“遮挡”住其显示。窗口置顶具有很大的意义，例如观看视频时就不希望正在后台运行的其他程序遮住播放的视频画面，某些程序会提供选项让用户决定是否要将窗口置顶，但并非每个应用程序都内置此功能，例如广泛应用的记事本 (Notepad) 和计算器 (calc) 就没有此功能，假设读者正在根据一个 TXT 文件中的数据进行计算，此时用户可能会从 TXT 文件中复制一些数据贴到计算器中，但一旦鼠标点击到记事本程序中，计算器程序就被覆盖，要重新激活其窗口就需要到任务栏点击，如果计算器程序窗口始终置顶，那么这项工作就可能变得容易许多。将介绍一个任意窗口置顶程序的设计与实现。

2　窗口置顶

事实上，窗口置顶只需要调用一个 Win32 API 函数 SetWindowPos，即可实现，以下是 MSDN 中关于此函数的原形签名：

```
BOOL SetWindowPos(HWND hWnd,
    HWND hWndInsertAfter,
    int X,
    int Y,
    int cx,
    int cy,
    UINT uFlags
);
```

其中第一个参数表示需设置窗口的句柄，第二个参数可通过传入 HWND_TOPMOST 或 HWND_NOTOPMOST 来指示置顶或取消置顶，第三到第六个参数在设置置顶或不置顶时没有实际意义，最后的参数可通过传入 SWP_NOMOVE 和 SWP_NOSIZE 来表明此次操作并不改变窗口的大小及位置而只是设置窗口置顶与否，因此，当需要置顶窗口时，可以使用下面的代码：

```
SetWindowPos (窗口句柄, HWND_TOPMOST, 1, 1, 1, 1, SWP_NOMOVE | SWP_NOSIZE);
```

如果需要取消置顶，那么只须调用下面的代码：

```
SetWindowPos(窗口句柄, HWND_NOTOPMOST, 1, 1, 1, 1, SWP_NOMOVE | SWP_NOSIZE);
```

3　获取任意窗口的窗口句柄

从上文中可以看到，实现窗体置顶与否的 API 函数及各项参数基本都已经获取，如果程序要实现的是对自己窗口的置顶，那么就非常容易，因为取得自己窗体的句柄是很容易实现的，但问题是现在的需求是实现对系统中存在的任意窗体置顶，因此上文调用 SetWindowPos 函数要传入的第一个参数应该是系统中任意窗体的句柄，所以问题的关键在于如何获取任意窗体的句柄，这就要使用另一个 Win32 API 函数 EnumWindows，下面详细分析。

首先来看 EnumWindows 在 MSDN 中的原形签名：

```
BOOL EnumWindows( WNDENUMPROC lpEnumFunc,
    LPARAM lParam
);
```

此函数用来枚举所有系统中的窗口 (实际上不包含属性为 WS_CHILD 的窗口)，其传入的第一个参数是一个回调函数的地址，即每当找到一个窗口时，回调函数都会执行，以下是回调函数的原形签名：

```
BOOL CALLBACK EnumWindowsProc( HWND hwnd,
    LPARAM lParam
);
```

此回调函数的第一个参数就是被枚举到的窗口的句柄，也就是我们需要的传入 SetWindowPos 函数的参数，第二个参数是调用 EnumWindows 函数时传入的第二个参数，用于向回调函数传入一些辅助信息。

4 筛选窗口

如果读者尝试着利用上文描述的 EnumWindows 函数来枚举一下自己计算机系统下的窗口的话就会发现，其返回的窗口数量非常之多，远远超过自己眼睛能一目了然看到的屏幕上的窗口，这并不矛盾，因为 Windows 系统及其复杂，除了表面看到的如“Windows Explorer”、IE 浏览器等窗口外，还有许多无法直接看到的窗口，如隐藏窗口、缩小在托盘区的窗口等，考虑到本程序的实际情况，对于这些无法直接看到的窗口，显然也没必要让其置顶，因此，为了筛选出尽量少的窗口供用户选择，可以考虑在枚举窗体的回调函数中进行特别处理，要完成甄别窗体的工作，需要用到另一个 Win32 API 函数 GetWindowInfo，其函数签名如下：

```
BOOL GetWindowInfo( HWND hwnd,
    PWINDOWINFO pwi
);
```

函数通过传入要甄别的窗口句柄 hwnd，返回一个 wndinfo 结构变量，该结构体的其中一个成员 dwStyle 是对窗口属性的描述，笔者考虑将窗口属性不含 WS_VISIBLE (即隐藏窗口) 和 WS_CAPTION (即含有标题栏) 的窗口都筛选出去，读者可根据实际情况做出修改。下面给出回调函数的示例代码，供读者参考。

```
        private bool EnumWindowsCallback (IntPtr handle,
IntPtr extraParameter)
        {
            int iWndLength = GetWindowTextLength (new
HandleRef(this, handle)) * 2;
            StringBuilder sb = new StringBuilder(iWndLength);
GetWindowText(new HandleRef(this, handle), sb, sb.Capacity);
      tagWINDOWINFO wndinfo = new tagWINDOWINFO();
     NativeMethods.GetWindowInfo(handle, ref wndinfo);
   if((wndinfo.dwStyle & NativeConstants.WS_VISIBLE) == 0)
                return true; //筛掉 Invisible 窗口
  if((wndinfo.dwStyle & NativeConstants.WS_CAPTION) == 0)
                return true; //筛掉无 titlebar 窗口
 // sb 变量包含筛选后得到的窗口的标题栏文字,此文字作为用
 // 户区分置顶哪个窗口的标志添加到 listbox 中
            listBox_wnd.Items.Add(sb.ToString());
 // 同步记录窗口的实际句柄(handle),以便 SetWindowPos 函
 // 数调用
            wndhandle[index++] = handle;
 // return true 表明继续枚举下一个窗口
            return true;
        }
```

5 C# 程序调用 Win32 API 的注意点

使用 C# 来实现此程序，如前文所介绍，程序中将使用许多 Win32 API 函数，C++语言通过引入头文件方式来声明 API 函数后直接调用之，而 C# 则必须使用 Dllimport 来导入具体实现 API 的 DLL，另外，Win32 API 函数广泛使用了许多结构体、常量等数据结构，在 C# 中，我们必须自己对这些结构进行声明，这无疑给编程者带来许多麻烦，好在已经有高手给我们准备好了相关工具，笔者使用了一款 P/Invoke 软件，具体地址读者可参考一下 url 或网上搜索，凭借该工具 (http://msdn.microsoft.com/en-us/magazine/cc164193.aspx)，读者只需将需要调用的 API 函数名或 API 常量名输入，就可以获取一段 C# 声明代码，随后编程者只须象调用.Net 类库函数那样直接调用 API 函数，以 GetWindowInfo 为例子，利用该工具生成的声明代码如下：

```
    [System.Runtime.InteropServices.StructLayoutAttribute
(System.Runtime.InteropServices.LayoutKind.Sequential)]
        public struct HWND__
        {
            /// int
            public int unused;
        }
[System.Runtime.InteropServices.StructLayoutAttribute
(System.Runtime.InteropServices.LayoutKind.Sequential)]
        public struct tagWINDOWINFO
        {
            /// DWORD->unsigned int
            public uint cbSize;
            /// RECT->tagRECT
            public tagRECT rcWindow;
            /// RECT->tagRECT
            public tagRECT rcClient;
            /// DWORD->unsigned int
            public uint dwStyle;
            /// DWORD->unsigned int
            public uint dwExStyle;
            /// DWORD->unsigned int
            public uint dwWindowStatus;
            /// UINT->unsigned int
            public uint cxWindowBorders;
            /// UINT->unsigned int
            public uint cyWindowBorders;
            /// ATOM->WORD->unsigned short
            public ushort atomWindowType;
            /// WORD->unsigned short
            public ushort wCreatorVersion;
        }
[System.Runtime.InteropServices.StructLayoutAttribute
(System.Runtime.InteropServices.LayoutKind.Sequential)]
        public struct tagRECT
        {
            /// LONG->int
            public int left;
            /// LONG->int
```

```
        public int top;
        /// LONG->int
        public int right;
        /// LONG->int
        public int bottom;
    }
    public partial class NativeMethods
    {
        /// Return Type: BOOL->int
        ///hwnd: HWND->HWND__*
        ///pwi: PWINDOWINFO->tagWINDOWINFO*
        [System.Runtime.InteropServices.DllImportAttribute ("
user32.dll", EntryPoint = "GetWindowInfo")]
      [return: System.Runtime.InteropServices.MarshalAsAttribu
te(System.Runtime.InteropServices.UnmanagedType.Bool)]
        public static extern bool GetWindowInfo ([System.
Runtime.InteropServices.InAttribute ()] System.IntPtr hwnd,
ref tagWINDOWINFO pwi);
        }
```

可以看到，该工具软件包括 API 函数中参数所使用的结构体等是有力工具，现在只须在代码中调用 NativeMethods.GetWindowInfo 就可以了，大大方便了编程者。

6 结语

通过 C# 语言并调用 Win32 API 实现了一个能将任意窗口置顶的工具，在进行多任务处理同时打开几个窗口时，可通过此程序对某个窗口进行置顶，以方便后续操作，该工具是对没有内置窗口置顶选项的应用程序的有力工具。读者可参考随附的源代码根据自己的需求进一步修改提高。

（收稿日期：2012-07-13）

（上接第 10 页）

```
  <p>您目前还没有好友！</p>
 {% endif %}
<h3>将您添加为好友的用户列表</h3>
 {% if from_friends %}
  <ul class="friends">
   {% for from_friend in from_friends %}
      <li class = "friend" ><a href = " {{from_friend.
get_absolute_url}}">{{from_friend.username}}</a></li>
    {% endfor %}
  </ul><br>
 {% else %}
  <p>目前还没有用户将您添加为好友！</p>
 {% endif %}
 <h3>好友最近发布的程序代码</h3>
 {% if snippet_list %}
  {% include "snippet_list.html" %}
 {% else %}
  <p>您的好友还没有发布过程序代码！</p>
 {% endif %}
{% endblock %}
```

视图函数 friend_list_page 的命名 url 访问入口在 urls.py 文件中添加，如下所示：

```
url(r'^friends/$', friend_list_page, name='friend_list_page'),
```

3.6 删除用户好友

如果用户 A 打算将另一个用户 B 从自己的好友列表中删除（单向删除，即只删除从用户 A 到用户 B 的好友关系，但从用户 B 到用户 A 的好友关系仍然存在），那么用户 A 就要首先进入用户 B 的个人页面，然后单击该页面中的“将该用户从您的好友列表中删除”链接来达到这一目的。实现删除用户好友功能的视图函数是 delete_friend，其定义如下所示：

```
@login_required
"""定义视图函数 delete_friend。"""
def delete_friend(request):
"""如果以 get 方法请求的数据中包含参数 fn(该参数的值用于指
定将要从当前登录用户好友列表中删除的好友的用户名信息)。"""
    if request.GET.has_key('fn'):
   fn = request.GET['fn']
   friend = get_object_or_404(User, username=fn)
   try:
    friendship = Friendship.objects.get(from_friend=request.
user, to_friend=friend)
    friendship.delete()
    msg = u'已经从您的好友列表中成功删除了用户%s！' %
 friend.username
   except:
    msg = u'未能从您的好友列表中成功删除用户%s！' %
friend.username
    request.user.message_set.create(message=msg)
   return HttpResponseRedirect(friend.get_absolute_url())
  else:
   raise Http404('参数错误！')
```

视图函数 delete_friend 的命名 url 访问入口在 urls.py 文件中添加，如下所示：

```
url(r'^friend/delete/$', delete_friend, name='delete_friend'),
```

这样，用户就可以通过 url 地址 http://localhost:8000/friend/delete/?fn=<用户名>从自己的好友列表中删除<用户名>指定的好友了。

至此，全部讲述了社会化程序代码共享系统中包含的主要功能的实现过程。由于该系统中包含的用户注册、用户登录、修改用户登录密码、重设用户登录密码、RSS 的生成以及程序代码标签等相关功能的实现比较简单，读者可以直接查阅附带的配套源代码进行学习和参考。

（收稿日期：2012-07-09）

基于.Net Remoting 和构建线程池实现分布式计算

王文举

摘　要：通过建立线程池和.Net Remoting 技术实现了一种分布式计算，介绍了.Net Remoting 分布式计算的原理、方法和代码实现。

关键词：.Net Remoting 技术；C# 语言；分布式计算

1　引言

.Net Remoting 分布式计算（远程处理）其实就是两个对象跨应用程序域进行通信的行为，如图 1 所示两个对象跨独立应用程序域的基本过程，客户端就是发起与远程对象交互的实体，而服务器端就是承载远程对象的软件代理。.NET 远程处理框架主要包括 4 个部分：代理、消息、信道和格式化程序。

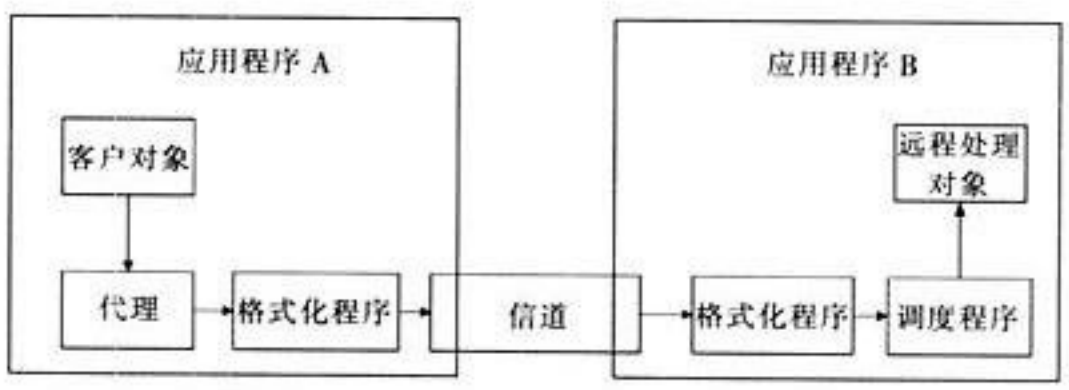


图 1　.Net 远程处理框架示意图

1.1　代理和消息

客户端和服务器端对象并不直接建立连接，而是通过一个称为代理的中间件。代理的作用就是“欺骗”客户端，让客户端相信它在和同一个应用程序域中的远程对象进行通信。代理提供统一的接口（例如成员、属性、字段等）来表现远程对象，客户端调用代理，而代理在后台把这些调用请求转交给远程对象。代理能够把客户端提供的变量格式化成消息对象，然后把消息对象传入信道。

1.2　格式化程序

TcpChannel 内置了一个格式化程序，它使用 BinaryFormatter 把消息转化成紧凑的二进制格式，数据包非常轻，远程访问快。格式化的消息生成后就被传入信道，最终它会抵达目标应用程序域。这个消息从符合协议的类型被格式化成符合.NET 的类型，远程对象的相应方法也同时被一个叫做调度程序的实体调用。

1.3　信道

信道是用来把消息传到远程对象的实体，局域网一般采用 TCP 信道，TCP 信道由 TcpChannel 类来表现，并使用 TCP/IP 网络协议传递消息。

2　设计思路

用.Net Remoting 平台构建分布式应用程序，一般需要 3 部分组成：普通程序集、客户端程序集、服务器端程序集。

2.1　构建普通程序集

首先构建普通程序集 MathLibrary.dll，它会被客户端和服务器端程序引用。MathLibrary.dll 中建立了一个 Remot_SimpleMath 类，它提供需要的计算方法，选择一个非常简单的自然数累加求和算法为例说明。Remot_SimpleMath 类是从 System.MarshalByRefObject 基类派生的，所以在客户端就可以通过代理进行访问。

2.2　构建服务器端程序集

服务器端程序集主要是用来承载包含远程对象的普通程序集的，新建一个 MathServer.exe 的控制台程序如图 2 所示。这个程序集的主要作用是为进入的请求打开信道，并通过 app.config 注册 Remot_SimpleMath。

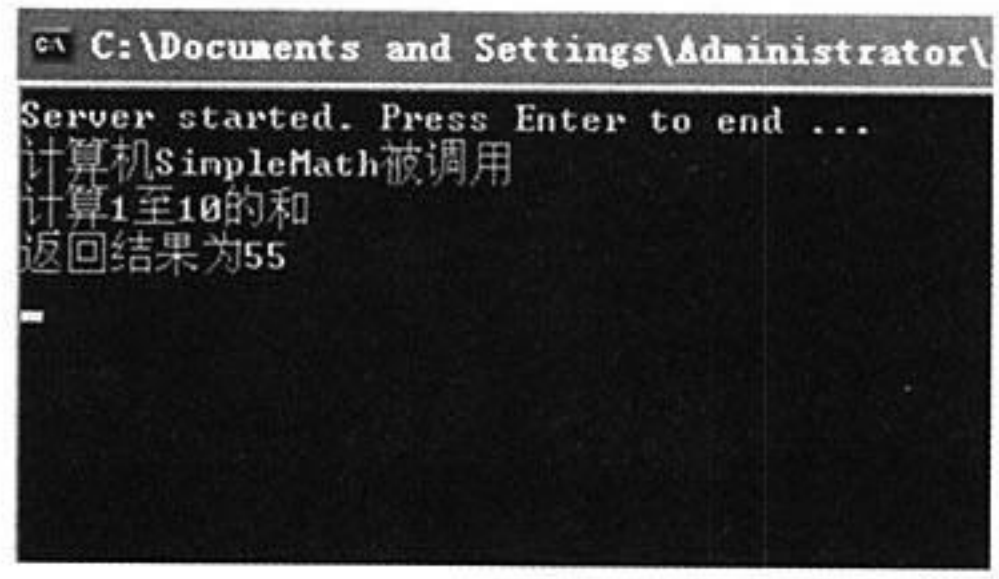


图 2　MathServer.exe 控制台程序

2.3　建立客户端程序集的代码

现在已经有了一个承载远程对象的监听器了，最后就是做一个请求服务的程序集。在应用中，通常是一个客户端要连接多个服务器，本文采取建立个线程池，让每个线程负责一个代

理，主程序只需和每个线程进行交互，线程的委托方法就是代理对象中的成员方法。为了实现线程池中每个线程的委托方法，文中建立一个线程委托方法类 Thread_SimpleMath，通过它把远端的代理和方法传递给线程。如图 3 所示，通过添加成员机的 IP、端口号，指定计算范围，在列表中添加成员机信息，并通过 ComputerData 结构体保存计算机信息。

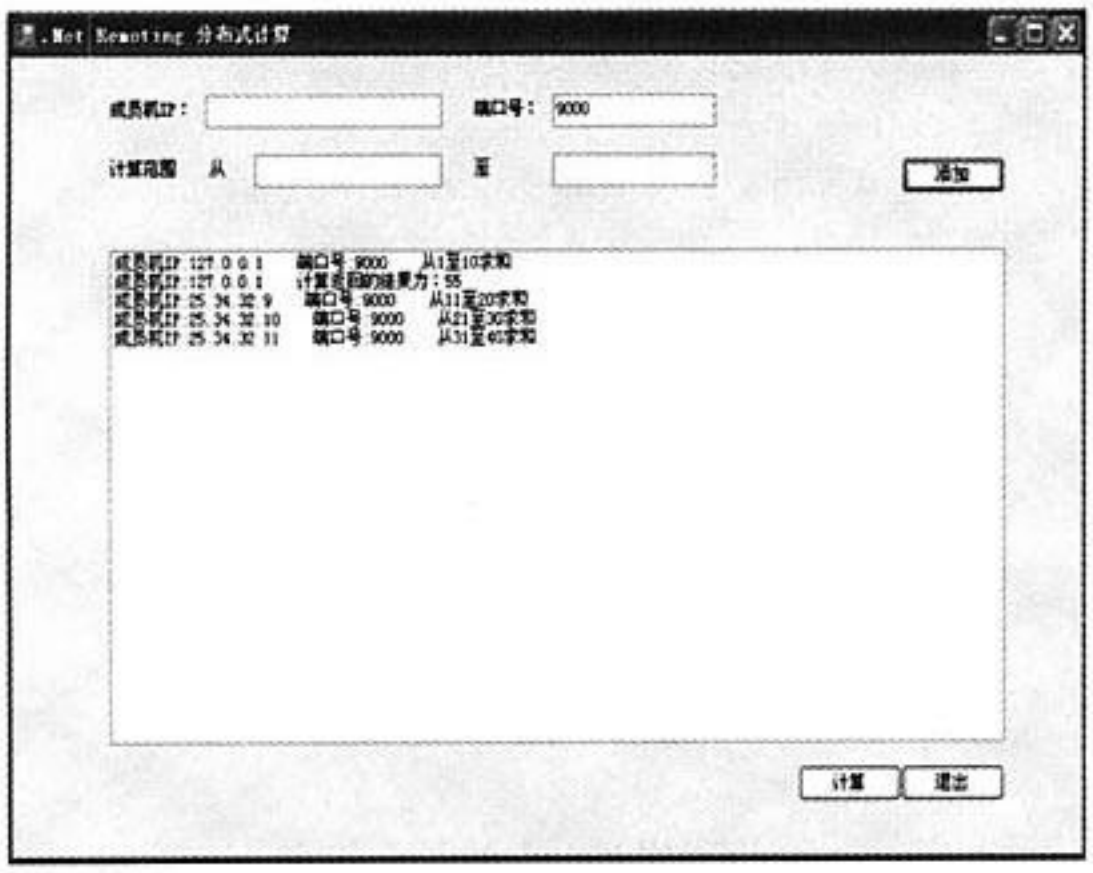


图 3　客户端程序界面

3　实现代码

3.1　构建普通程序

```
namespace MathLibrary
{   public class Remot_SimpleMath : MarshalByRefObject
    {   public Remot_SimpleMath()
        {  Console.WriteLine("计算机 SimpleMath 被调用"); }
        public int Add(int n1, int n2)
        {   Console.WriteLine("计算{0}至{1}的和", n1, n2);
            int result = 0;
            for (int n = n1; n <= n2; n++)
            { result += n;}
            Console.WriteLine("返回结果为{0}", result);
            return result;
        }
    }
}
```

3.2　构建服务器端程序

```
namespace MathServer
{
     class ServerMain
     {
    static void Main(string[] args)
    {
        RemotingConfiguration.Configure   ("MathServer.exe.config",false );
        // Keep the server alive until enter is pressed.
        Console.WriteLine("Server started. Press Enter to end ...");
        Console.ReadLine();
    }
     }
}
```

服务器端 app.config 文件代码如下：

```
<configuration>
  <system.runtime.remoting>
  <customErrors mode="Off"/>
   <application name="ServerLoader">
    <service>
      <wellknown mode="Singleton"
                          type="MathLibrary.Remot_SimpleMath,MathLibrary"
                          objectUri="Remot_SimpleMath"/>
    </service>
    <channels>
     <channel port="9000"
           ref="tcp server" />
  </channels>
   </application>
  </system.runtime.remoting>
</configuration>
```

3.3　建立客户端程序

```
namespace FDC
{
   public partial class ParallelSet : Form
   {
      public struct ComputerData
      {
         public string ip;
         public string port;
         public int from;
         public int end;
      }
      ComputerData re_computer; //成员机信息
      public ArrayList computerArray = new ArrayList();
      public Remot_SimpleMath remot_SMath; //远端代理
      Thread_SimpleMath[] thread_SMath; //线程委托方法
      public Thread[] computer_thread;//线程池
            private void add_button_Click (object sender, EventArgs e)
      {
         ComputerData computer1;
         computer1.ip = ip_textBox.Text.ToString();
         computer1.port = port_textBox.Text.ToString();
         computer1.from  = Convert.ToInt32 (from_textBox.
```

```
Text);
            computer1.end = Convert.ToInt32 (end_textBox.
Text);
            computerArray.Add(computer1);
            parallel_listBox.Items.Add("成员机 IP:" + ip_textBox.
Text.ToString () + "  端口号:" + port_textBox.Text.ToString()
+ "  从 " + from_textBox.Text.ToString () + " 至 " +
end_textBox.Text.ToString()+"求和");
        }
        private void calculate_button_Click (object sender,
EventArgs e)
        {
            //做个线程池,用于分布式计算
            thread_SMath = new Thread_SimpleMath
[computerArray.Count];
            computer_thread = new Thread [computerArray.
Count];
            for (int i = 0; i < computerArray.Count; i++)
            {
                re_computer.ip = ((ComputerData)computerArray
[i]).ip;
                re_computer.port = ((ComputerData)
computerArray[i]).port;
                re_computer.from = ((ComputerData)
computerArray[i]).from;
                re_computer.end = ((ComputerData)
computerArray[i]).end;
                try
                {string URL = "tcp://" + re_computer.ip + ":" +
re_computer.port + "/ServerLoader/Remot_SimpleMath";
                    //获得远端代理
remot_SMath = (Remot_SimpleMath)Activator.GetObject(
                typeof(Remot_SimpleMath), URL);
                    //远端代理和计算信息交给线程委托方法类
                thread_SMath [i] = new Thread_SimpleMath
(re_computer.from, re_computer.end, remot_SMath);
                //远程代理的方法作为线程的委托方法
                ThreadStart worker1 = new ThreadStart
(thread_SMath[i].TreadMethod);
                computer_thread[i] = new Thread(worker1);
//线程的建立
                computer_thread[i].Start();//开启所有线程
            }
            catch (Exception e1)
            { MessageBox.Show(e1.ToString());}
        }
        int[] result = new int[computerArray.Count];
        int num = 0;
        do
        {//返回每个远端计算机的计算结果
            for (int i = 0; i < computerArray.Count; i++)
            {
                if (result[i] ! = thread_SMath[i].result)
                {
                    result[i] = thread_SMath[i].result;
                    parallel_listBox.Items.Add (" 成员机 IP:" +
((ComputerData)computerArray[i]).ip.ToString() + "  计算返回
的结果为:" + result[i].ToString());
                    num++;
                }
            } while (num ! = computerArray.Count); //所有线程
//计算完毕
            MessageBox.Show("计算完毕! ");
        }
    }
}
//线程委托方法类
namespace FDC
{
    using MathLibray;
    class Thread_SimpleMath
    {
        Remot_SimpleMath remot_SMath;
        int from; int end;
        public int result = 0;
        public Thread_SimpleMath (int _from, int _end,
Remot_SimpleMath _remot_SMath)
        {
            from = _from;
            end = _end;
            remot_SMath = _remot_SMath;
        }
        public void TreadMethod()
        { result = remot_SMath.Add (from, end); }
    }
}
```

4 结语

通过建立线程池和线程委托方法类，每个线程负责一个代理，线程的委托方法就是代理对象中的成员方法，主程序只需和每个线程进行交互，实现了.Net Remoting 分布式计算。该方法充分利用局域网的计算资源，可以把非常巨大的计算能力问题分成许多小的部分，然后把它们分配给许多计算机进行处理，最后把这些结果综合起来得到最终的结果。

（收稿日期：2012-08-08）

基于 C# 的栈操作可视化演示的程序实现

孙义欣

摘　要：利用 C# 编程对顺序栈常见的基本操作进行了可视化演示，有助于编程爱好者掌握并理解这一数据结构的特点和使用。

关键词：栈；标签；动态；可视化；定时器控件

1　引言

栈是软件设计中常用的一种数据结构，在很多典型问题的求解中起着相当重要的作用。栈广泛应用于操作系统、编译原理、各种应用系统软件以及程序设计中，因而掌握栈的常用操作及其基本应用是解决很多实际问题的基础。

栈是一种特殊的线性表，其特殊性在于它的插入和删除操作时受限的，即限定仅在表尾进行插入和删除操作的线性表，即栈的修改是按照后进先出的原则进行的，因此，栈又称为后进先出表（Last In First Out，简称 LIFO）。栈的表尾端具有特殊的含义，称为栈顶，栈顶的第一个元素称为栈顶元素，表头端称为栈底，不含任何数据元素的栈称为空栈。元素进栈时从表尾依次进入，出栈时也是从表尾依次出栈。

通常，栈的实现有两种：顺序栈和链栈。利用顺序存储的方式实现的栈称为顺序栈，栈中的数据用一个一维数组来实现，栈底位置可以设置在数组的任意一个端点，栈顶随着插入和删除操作的进行而不断变化，用一个变量 top 作为指向栈顶的指针，指明当前栈顶的位置。通常，将栈的底部位置设置为 0，这样，当空栈时栈顶指针为-1（s.top= -1），表示栈中还没有数据；入栈操作时，首先让栈顶指针加 1（s.top++），在栈中空出一个元素的空间，然后将要入栈的元素放入栈中空出的区域；出栈时相反，首先将栈中栈顶指针指示的元素取出，然后栈顶指针减 1（s.top --），使栈顶指针下移一个位置，以便对下一个元素进行访问。

栈常见的基本操作主要有栈初始化、空栈、入栈和出栈等。栈操作示意图如图 1 所示，其中，（1）为空栈；（2）为进栈一个元素的状态及栈顶指针所指的位置情况；（3）为进栈 4 个元素后的状态及栈顶指针所指的位置情况；（4）为出栈一个元素后栈的状态及栈顶指针位置的情况。在示意图（4）中虽然仍有 4 个元素，但栈顶指针 s.top 已经不指向元素 18，所以元素 18 已无法访问，如果再有入栈的元素，就将占用元素 18 的区域，从而将元素 18 覆盖。

栈和队列这两种数据结构在程序设计中的作用非常重要，一般数据结构的教材中都是以图示结合文字说明的方式描述其操作，由于其操作的抽象性和特殊性，而且图示都是静态的，初学者不容易很好地理解和掌握。利用 C# 编程实现了栈操作的动态可视化演示，希望可以对初学者起到一定的帮助作用。

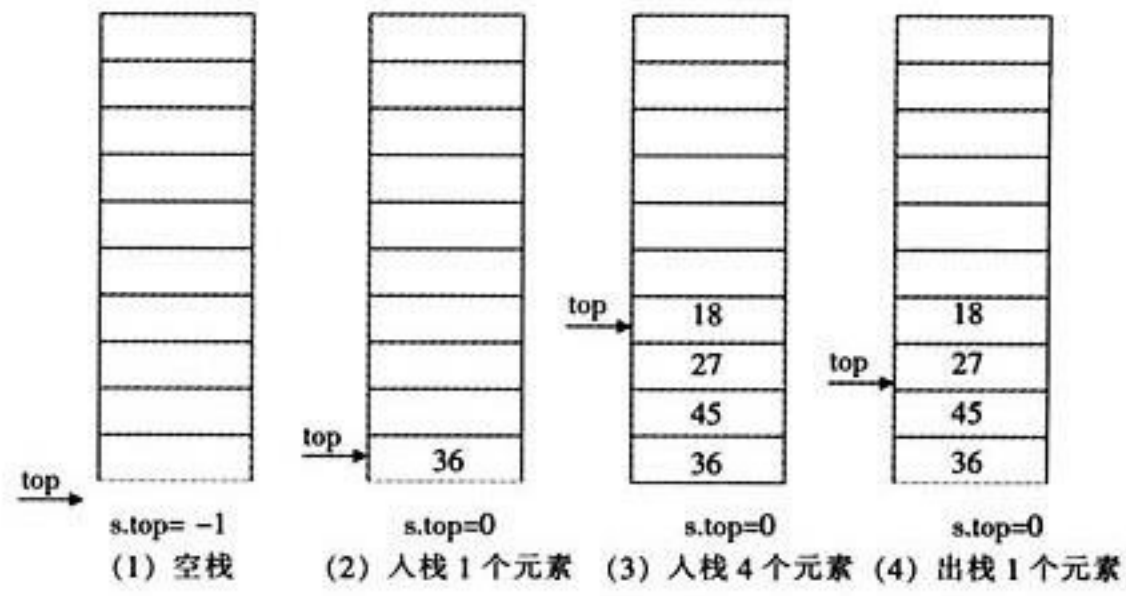


图 1　栈操作示意图

2　顺序栈操作可视化的程序实现

新建一个 Visual C# Windows 应用程序项目，项目名称为 StackDemoSln；将默认窗体名称 Form1 改为 FrmMainDemo；在窗体上添加 4 个 Button 按钮，其 name 属性分别为：btnStart、btnEmpty、btnPush、btnPop，其 Text 属性分别为：初始化、空栈、入栈和出栈；添加 4 个 Timer 控件，其 name 属性分别为 tmrPush1、tmrPush2、tmrPop1、tmrPop2，其 Interval 属性都定义为 200，程序中利用 Timer 控件的定时功能代替循环以达到动态演示数据移动的效果。程序运行时当点击“初始化”按钮时，会动态生成一组标签控件作为顺序栈操作演示的平台。程序中的主要代码如下：

```
//类中定义的主要数据如下
private Label[] lblStackData = new Label[10]; //栈区域
private Label lblSource;   //用于存放即将入栈的数据
private Label lblTarget;   //用于存放出栈后的数据
private Label lblMoveData; //移动的数据
```

```
    private Label lblPointer;  //栈顶指针指示器
    private Random random;     //随机数发生器,用来随机产
//生入栈的数据
    private int number;
    private int position = 0; //栈中数据个数统计
    private int xPos, yPos;   //记录移动位置
    //初始化按钮的单击事件,用来生成初始的操作界面
  private void btnStart_Click(object sender, EventArgs e)
  { //动态生成将要入栈的数据并在标签中显示
    lblSource = new Label();
    lblSource.Size = new Size(60, 30);
    lblSource.Top = 20;
    lblSource.Left = 20;
    lblSource.BorderStyle = BorderStyle.FixedSingle;
    lblSource.TextAlign = ContentAlignment.MiddleCenter;
    this.Controls.Add(lblSource);
    //动态生成用来存放出栈数据的标签
    lblTarget = new Label();
    lblTarget.Size = new Size(60, 30);
    lblTarget.Top = 20;
    lblTarget.Left = 400;
    lblTarget.BorderStyle = BorderStyle.FixedSingle;
    lblTarget.TextAlign = ContentAlignment.MiddleCenter;
    this.Controls.Add(lblTarget);
    //生成移动标签
    lblMoveData = new Label();
    lblMoveData.Size = new Size(60, 30);
    lblMoveData.ForeColor = Color.White;
    lblMoveData.BackColor = Color.Blue;
    lblMoveData.Top = 20;
    lblMoveData.Left = 20;
    lblMoveData.BorderStyle = BorderStyle.Fixed3D;
                lblMoveData.TextAlign = ContentAlignment.
MiddleCenter;
    this.Controls.Add(lblMoveData);
    //动态生成栈区域,最多存放 10 个数据
    for (int i = 0; i < 10; i++)
    {   lblStackData[i] = new Label();
        lblStackData[i].Size = new Size(60, 30);
        lblStackData[i].Top = i * 30 + 110;
        lblStackData[i].Left = 200;
        lblStackData[i].BorderStyle = BorderStyle.FixedSingle;
          lblStackData [i].TextAlign = ContentAlignment.
MiddleCenter;
        this.Controls.Add(lblStackData[i]);
    }
     //生成栈指针指示器,始终指向栈顶位置,初始时指向栈的
//底部
    lblPointer = new Label();
    lblPointer.Size = new Size(60, 30);
    lblPointer.Top = 410;
    lblPointer.Left = 160;
    lblPointer.Text = "—>";
    this.Controls.Add(lblPointer);
    //随机产生入栈数据
    random = new Random();
    number = random.Next(100);
    lblSource.Text = number.ToString();
    lblMoveData.Text = lblSource.Text;
    btnPop.Enabled = false;
    btnPush.Enabled = true;
     btnStart.Enabled = false;
  }
   //设置为空栈,栈顶指针指向栈的底部,出栈操作无效
   private void btnEmpty_Click(object sender, EventArgs e)
   { lblPointer.Top = 410;
     position = 0;
     btnPush.Enabled = true;
     btnPop.Enabled = false;
   }
   //入栈操作
    private void btnPush_Click(object sender, EventArgs e)
   { if (position >=10)       //判断栈是否已满
     {   MessageBox.Show("栈已满！", "Message");
         btnPush.Enabled = false;
         position = 10;
     }
     else
     {  lblMoveData.Visible = true;
           lblMoveData.Text = lblSource.Text;
        lblMoveData.Left = 20;
        xPos = 20;
        lblMoveData.Top = 20;
        yPos = 20;
        lblPointer.Top -= 30;
        tmrPush1.Enabled = true;  //启动入栈数据移动
      }
    }
   //入栈操作:数据横向移动
    private void tmrPush1_Tick(object sender, EventArgs e)
   {   xPos += 60;
       if (xPos <= 200)
       {  lblMoveData.Left = xPos;     }
       else
       {   tmrPush1.Enabled = false;
           tmrPush2.Enabled = true;  //横向移动完成,启动向下移
   //动进入栈区域
       }
   }
   //入栈操作:数据纵向移动
   private void tmrPush2_Tick(object sender, EventArgs e)
   {   yPos+=30;
       if (yPos <= 380 - position * 30)
       {   lblMoveData.Top = yPos;     }
```

```
else
{   tmrPush2.Enabled = false;
    lblStackData[9 – position].Text = lblMoveData.Text;
    lblMoveData.Visible = false; ;
    btnPop.Enabled = true;
    position++;
    number = random.Next(100);
    lblSource.Text = number.ToString();
}
//出栈操作
private void btnPop_Click(object sender, EventArgs e)
{   position––;
   if (position <0)   //判断栈是否已空
   {   MessageBox.Show("栈已空！ ", "Message");
       btnPop.Enabled = false;
       position = 0;
   }
   else
   {   lblMoveData.Top = (9 – position) * 30 + 110;
       yPos = lblMoveData.Top;
       lblMoveData.Left = 200;
       xPos = 200;
       lblMoveData.Text = lblStackData[9 – position].Text;
       lblMoveData.Visible = true;
       tmrPop1.Enabled = true;  //出栈数据开始移动
   }
}
//出栈操作:数据向上移动,逐渐离开栈区域
private void tmrPop1_Tick(object sender, EventArgs e)
{   yPos –= 30;
    if (yPos >= 20)
    {  lblMoveData.Top = yPos;    }
    else
    {   tmrPop1.Enabled = false;
          tmrPop2.Enabled = true; //向上移动完成,启动横向移
//动,出栈数据进入目的区
    }
}
//出栈操作:数据横向移动
private void tmrPop2_Tick(object sender, EventArgs e)
{   xPos += 60;
    if (xPos < 400)
    { lblMoveData.Left = xPos;    }
    else
    {   tmrPop2.Enabled = false;
        lblTarget.Text = lblMoveData.Text;
        lblPointer.Top += 30;
        btnPush.Enabled = true;
        lblMoveData.Visible = false;
    }
}
```

3 程序运行界面

程序运行画面如图 2 所示。程序运行时，当单击“初始化”按钮时，动态生成初始的操作演示界面；单击“入栈”按钮时，首先栈顶指针指示器上移一个位置，然后数据从左上角的源数据位置逐渐进入栈区域中栈顶指针指示的位置；单击“出栈”按钮时，数据从栈区域中栈顶指针指示的位置移出，进入右上角的目标位置，然后栈顶指针下移一个位置；单击“空栈”按钮时，栈顶指针指向栈的底部。入栈和出栈操作中的数据移动是动态进行的，非常直观、形象。

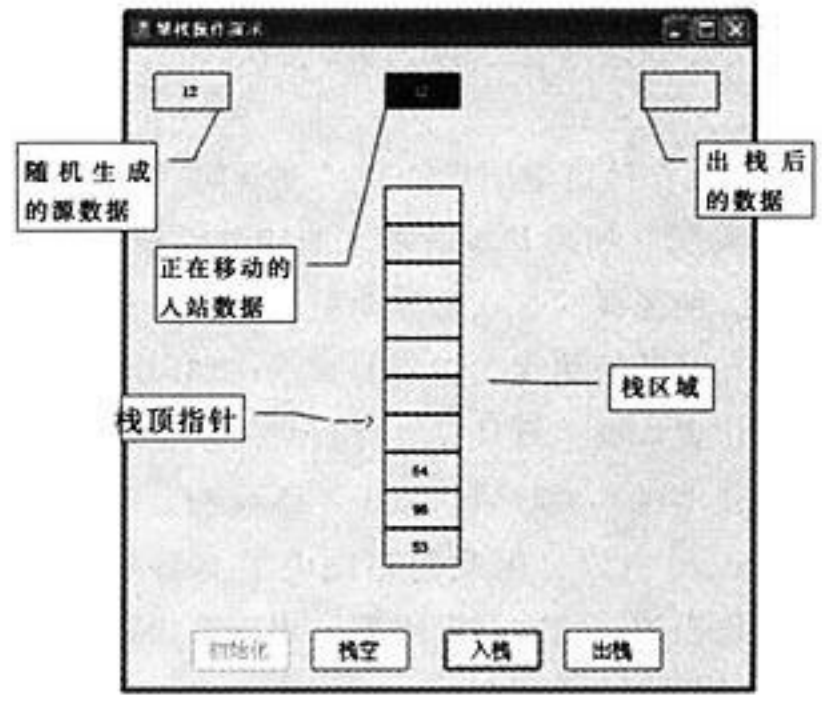


图 2 程序运行画面

4 结语

栈在程序设计中的作用相当重要，而栈的操作又比较特殊，初学者在理解和使用时感觉比较抽象，难以很好的理解，在编程中也不能自如地使用。文中用 C# 编程实现了顺序栈操作的可视化演示，将栈抽象的常用操作进行了直观的动态展示，有助于编程爱好者更好地理解和掌握这一重要数据结构的特点。

参考文献

[1] 严蔚敏，吴伟民. 数据结构（C 语言版）[M]. 北京：清华大学出版社，1997.

[2] 邓文华，李益明，等. 数据结构（C 语言版）[M]. 北京：电子工业出版社，2007.

[3] 尹立宏. Visual C#.NET 应用编程 150 例 [M]. 北京：电子工业出版社，2003.

[4] 林邦杰. 深入浅出 C# 程序设计 [M]. 北京：中国铁道出版社，2005.

（收稿日期：2012-09-05）

二维 Bezier 曲线求交算法及其比较

朱根荣

摘　要：用扫描法、两分查找法、牛顿法、离散法、代数法求 Bezier 曲线交点的算法思想，及在 Adobe ActionScript 3.0 中的实现，并指出了存在问题，给出了改进办法。通过实验比较，解非线性方程组法是诸方法中效率最高、稳定性最好的方法。

关键词：Bezier 曲线；交点；算法

在用 Flash 制作经济学图形的演示动画时，经常遇到求两条曲线交点的问题。如最基本的需求曲线与供给曲线的交点。当然在设计时，两曲线的交点是显而易见的。困难的是在动画运行期间，正在移动的曲线，它们的交点该如何确定。这不仅需要所求交点位置准确，而且需要有较快的求交速度。一旦速度太慢，必然使动画产生时滞、显得不够流畅。鉴于设计时所画曲线均为 Bezier 曲线，因此问题便成了 Bezier 曲线求交问题。曹锋曾在发表的一篇文章中提到了用解高次方程组的方法求 Bezier 曲线的交点，并不简单。对于求两条三次 Bezier 曲线交点的问题，最终要转化为解关于参数的 9 次方程，不得不借助代数方程的数值解法。没有尝试，但受其启发，最终用“牛顿法解非线性方程组”实现了这一算法。王国瑾在其《计算机辅助几何设计》中介绍了化曲为直的离散未交算法，在 AS3.0 中实现后，效果不错。此外，还尝试了扫描法、两分查找法、牛顿法。在此对各种算法及其在 AS3.0 中的实现、存在问题、改进办法作一归纳总结，并通过实验比较证明：用牛顿法解非线性方程组的方法求两 Bezier 曲线交点，效率最高，也最稳定。

1　二维 Bezier 曲线的参数表示

在曲线运动过程中求交点，无疑需要借助有关算法在动作脚本中以代码实现。而要用代码求两 Bezier 曲线的交点，首先需要将曲线表示成函数。但对 Bezier 曲线而言，要用普通函数来表示是困难的，幸可用参数方程。当给定平面上 n+1 个点 d_i (i=0,1……n)，则 n 次 Bezier 曲线可表示为 $P(t)=\sum_{i=0}^{n} C_n^i t^i (1-t)^{n-i} d_i$。

$0 \le t \le 1$ 其中 C_n^i 是组合数，即 $C_n^i = \frac{n!}{i!(n-i)!}$，而 P (t) 是参数为 t 时曲线的位置，是一个二维矢量。容易看出方程右边实际上是一个 n 次多项式，可整理成 $P(t)=\sum_{i=0}^{n} a_i t^i$，$0 \le t \le 1$，这里系数 a_i 是一个二维矢量。

鉴于 AS3.0 中无法重载运算符，因而无法定义矢量运算类，需将矢量方程化成标量方程，形如：$\begin{cases} x(t)=\sum_{i=0}^{n} a_i t^i & \cdots\cdots(1) \\ y(t)=\sum_{i=0}^{n} b_i t^i & \cdots\cdots(2) \end{cases}$

其中 a_i、b_i 为标量系数，可从给定控制点 d_i 求得[1]。对于三次 Bezier 曲线，两参数方程的系数 $\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = M_B \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$，$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = M_B \cdot \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}$，其中 (x_i,y_i) 为控制点 di 的横、纵坐标。M_B 为三次 Bezier 曲线的 4x4 矩阵。下面给出根据控制点求参数方程系数的 AS3.0 代码如下：

```
//代码 1：根据 Bezier 曲线的控制点求参数方程的系数
    //参数：ds 为 Bezier 曲线控制点数组
    //返回：参数方程系数数组：a0,a1,…an;b0,b1…bn
  function getCoefs(ds:Array):Array {
        var n:int;
            var M:Array;
            var coefs:Array=new Array(2);
            coefs[0]=new Array(n+1);
            coefs[1]=new Array(n+1);
            n=ds.length;
            M=new Array(n);
            switch(n){
                case 2:
                    M[0]=[1,0];
                    M[1]=[-1,1];
                    break;
                case 3:
                    M[0]=[1,0,0];
                    M[1]=[-2,2,0];
                    M[2]=[1,-2,1];
                    break;
                case 4:
                    M[0]=[1,0,0,0];
                    M[1]=[-3,3,0,0];
                    M[2]=[3,-6,3,0];
```

```
                M[3]=[-1,3,-3,1];
                break;
        }
        for(var i:int=0;i<n;i++){
            coefs[0][i]=0;
            coefs[1][i]=0;
            for(var j:int=0;j<n;j++){
                coefs[0][i]+=(M[i][j]*ds[j].x);
                coefs[1][i]+=(M[i][j]*ds[j].y)
            }
        }
        return coefs;
    }
```

2 二维 Bezier 曲线求交算法及其优化

2.1 扫描法 (步进法)

2.1.1 算法与代码

扫描法是最先想到的求交方法，基本思路是用一条与横轴垂直的直线 (扫描线) 从两曲线相交区域的左端向右端扫描，扫描线每到一个位置 x，以便计算两曲线纵坐标之差的绝对值 dy=|y1-y2| (如图 1) ，当 dy 小于给定的精度 ε (一个小正数)，便将点 (x, (y1+y2) /2) 近似为两曲线的一个交点。代码如下：

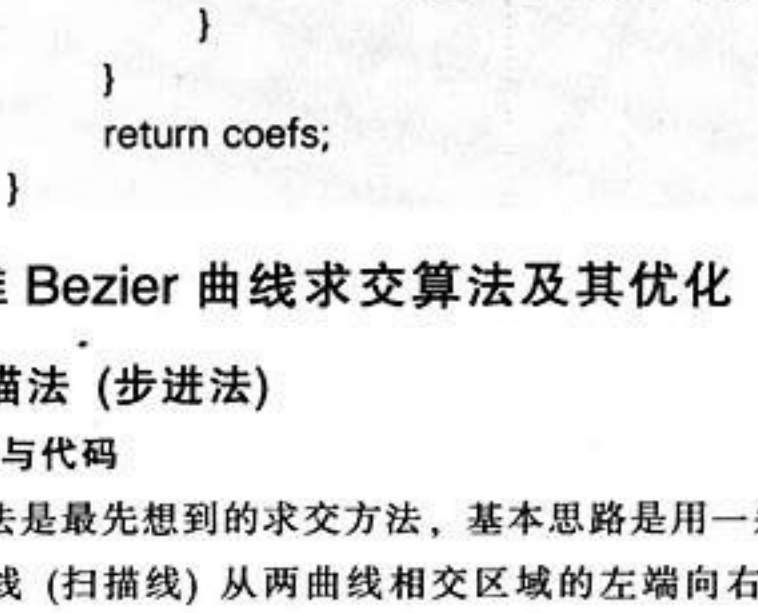


图 1 扫描法求交点的算法描述

```
//代码 2:扫描法求交点的基本代码
//参数:ca、cb 表示两曲线,系自定义 Bezier 曲线类实例
//返回:交点数组
function scan(ca:McCurve,cb:McCurve):Array{
    var left,right:Number;      //两曲线相交区域的左、右端
    var y1,y2:Number;    //两曲线在 x 时的纵坐标
    var ,dy:Number;      //两曲线在 x 时纵坐标差的绝对值
    var step:Number=1; //步长
    var eps:Number=1;  //精度
    var pts:Array;                  //存放交点的数组
    for(x=left;x<=right;x+=step){
        y1=ca.getYByX(x);
        y2=cb.getYByX(x);
        dy=Math.abs(y1-y2);
        if(dy<eps){
            pt=new Point(x,(y1+y2)/2);
            pts.push(pt);
        }
    }
    return pts
}
```

为了突出算法重点，代码中没有给出两曲线相交区域左、右端的计算过程。这在 ActionScript3.0 中不难解决，可用 MovieClip 类的 getBounds ()、getRect () 函数获得两曲线的包围盒，再用 Rectangle 类的 intersection () 方法求出两包围盒的相交区域 (不妨记为 rect)。于是相交区域的左、右端即为：

```
left=rect.x;
right=rect.x+rect.width;
```

2.1.2 存在问题

该算法虽然简单明了，但实现时问题不少。

(1) 根据 x 值求 y 值的 getYByX (x) 函数的实现问题

由 Bezier 曲线的参数方程可知，要根据 x 值求 y 值，需分两步：

①先从(1)式中由给定的 x 值求出参数 t 值；

②再从(2)式中根据 t 值求出 y 值。

对于三次 Bezier 曲线，第①步要解关于 t 的一元三次方程，并不简单，所幸的是 ActionScript3.0 的 BezierSegment 类提供了根据方程系数求三次方根的函数 getCubicRoota (a,b,c,d)。对于更高阶的 Bezier 曲线，就没有现存方法可用，只有用代数方程的数值解法求得近似解。

(2) 无值、多值问题

上面第①步从方程(1)中求 t 值时，理论上有 n 个根。剔除 [0,1] 以外的值后，t 值的个数在 0 与 n 之间，事先无法确定。相应地第②步得到的 y 值的个数与 t 值个数相同，也在 0 与 n 之间。因此，在计算 dy=|y1-y2|时，要根据 y 值的个数作出相应的处理。如果 y1 或 y2 的个数有一个为 0，则回到循环开始进行下一循环；否则，需对每个 y1 值与每个 y2 值两两求差取绝对值，再行判断，需用到双循环。

```
var ys1,ys2:Array;              //两曲线在横坐标为 x 时的纵坐标
//值数组
var i,j:int;
for(i=0;i<ys1.length;i++){
    for(j=0;j<ys2.length;j++){
        dy=Math.abs(ys1[i]-ys2[j]);
        if(dy<eps){
            pts.push(new Point(x,(ys1[i]+ys2[j])/2);
        }
    }
}
```

(3) 步长、精确度与交点个数

当步长 step 较大时，运行速度较快，但会发生交点漏失，

即明明相交的地方，因步长较大被跨过，而未被检测到，使求得的交点个数小于实际交点数。

当步长 step 很小时，虽可能避免交点漏失的情况，但会带来两个问题，一是运行速度减慢，二会使交点个数虚增。即在一个交点附近，多次被判断为相交，使求得的交点个数大于实际交点数。而若减小精确度，又可能导致交点漏失。

2.1.3 改进思路

对于问题(1)，曾经想到将“对 x 值扫描”改为“对参数 t 扫描”，这样根据 t 值求相应的 y 值时，无须解 n 次方程，只需将 t 值代入方程(2)即可求得，且也避免了多值问题。但这仅仅得到了一条曲线的纵坐标值。由于 t 值相同时，两条曲线的横坐标值未必相同，因此另一条曲线的纵坐标值不能简单地用相同的参数 t 值来求，而要根据一条曲线的 x 值求出另一条曲线的参数值 (为了区别第一条曲线的参数，用 s 表示)，再由 s 求出另一条曲线的 y 值。

这里，根据一条曲线的 x 值求另一条曲线的参数值 s 还得解 n 次方程，还会有多个 y 值。改进效果不明显，放弃。

对于问题(2)，既对参数 t 扫描仍然无法彻底解决多值问题，又尝试了对纵坐标 y 扫描。一般来说，给定 x 求 y 值时有多个值时，反过来给定 y 值求 x 值，多值问题会有所改善，但仍然无法避免。比如，当曲线为类似圆时，无论前者还是后者，都会有两个值。因此，考虑到兼容性，针对多值问题的这个双循环是免不了的了。

对于问题(3)，想到的是在扫描过程中，改变步长。即在不太可能相交时，步长大一点，在可能相交时，步长小一点。前者可以提高速度，后者可以避免交点漏失。想法虽好，但问题是，如何判断某一时刻是不太可能相交，还是将要相交，不太可能相交是，步长取多大，快要相交时步长又该取多大。一个简单的办法时，根据 dy (在横向扫描时) 的大小来决定步长。在代码 2 中计算出 dy 后，添加以下代码：

```
if(dy<eps) {
    //添加点到结果数组
    step=2;                    //这是为了避免交点数虚增,但在两
//曲线接近相切,两交点很近 时,
   //也可能导致交点漏失
}
else if(dy<10) step=0.1*dy;     //dy 较小时,将要相交,步长减
//小
else step=2;                    //否则取较大的步长
```

这一办法，虽然对求交速度和准确性有所改善，但改善不大。为此又尝试了用两曲线的斜率、二阶导数、曲率作为决定下一步步长的依据，但都无功而返。最后采取的办法是，求扫描线处两曲线切线的交点，比较切线交点与扫描线的有向距离来判断两曲线的相交趋势，进而决定步长的大小。如图 2 中曲线 A 的切线 LA 与曲线 B 的切线 LB 相交于点 P，点 P 与扫描线 X 的距离 d 即是判断相交趋势并决定步长大小的依据。图 2 中是即将相交的情形，可取步长为 d 乘以一个小于 1 的正数。如果 P 在扫描线的左边，则说明暂时不会相交，步长可适当取大一些。当然在具体实现中，还有很多特殊情况需要考虑，在此，就不一一细说了。

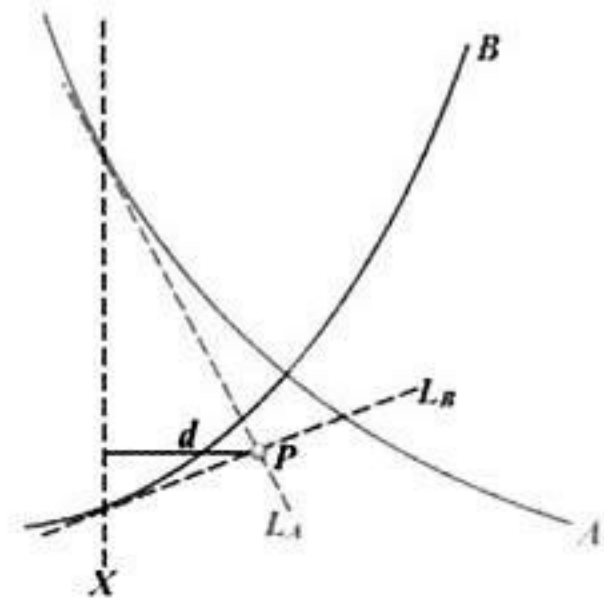


图 2 扫描法可变步长的确定

这样一来，虽然增加了求曲线切线及其切线交点的运算，但可大大减少扫描的次数，实验证明，对提高求交速度和交点数量的准确性还是有一定效果的。

2.2 两分 (折半) 查找法

2.2.1 算法与代码

两分查找法是一种十分常用的算法，通常用于在一个排序数组中寻找某个元素。笔者将其用于曲线求交，得归功于《数学手册》中用两分查找法解代数方程的启示。若设两曲线相交区域的左、右端为 L、R (见 1.1.1)，则两曲线交点的求解过程是：

①在区间 [L,R] 中插入中点 M，将原区间分成左、右两个子区间 [L,M] 和 [M,R] (如图 3)。

②分别计算 x=L、M、R 时两曲线 y 值之差 (不是绝对值)，记为 ldy、mdy、rdy，则左子区间两端两曲线 y 值之差为 ldy，mdy；右子区间两端两曲线 y 值之差为 mdy，rdy。

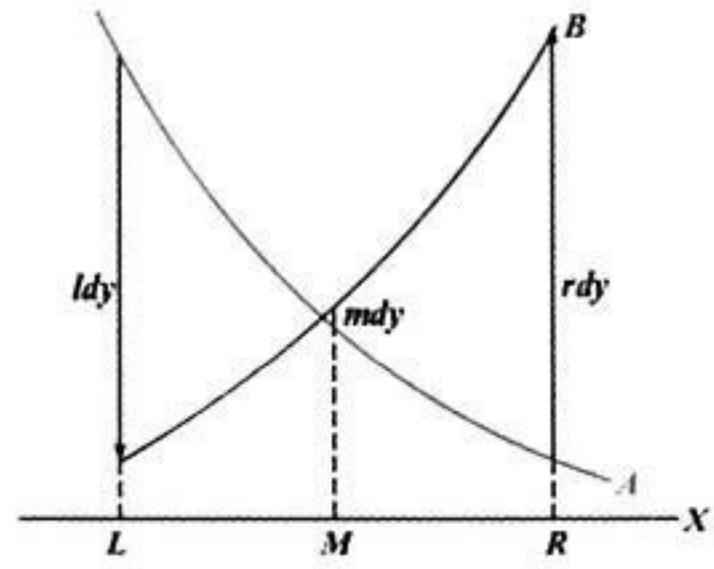


图 3 两分查找法求交点算法描述

③若某子区间两端 y 值之差同号，则直接无视，舍去这一子区间 (图 3 中 [M,R])。若某子区间两端 y 值之差异号 (图 3

中 [L,M])，在这一子区间中再插入中点 M，将此子区间分成两段，重复②，③，直到子区间的长度小于给定精度 ε，或者|ldy|、|mdy|、|rdy|中有一个小于 ε 为止。若是|ldy|<ε，则将 L 作为交点的横坐标，x=L 时两 y 值的平均值作为交点的纵坐标；若是|mdy|或|rdy|<ε，则将 M 或 R 作为交点的横坐标，相应两 y 值的平均值作为交点的纵坐标。

```
//代码 3:两分查找求交的循环算法
function binary(ca:McCurve,cb:McCurve):Point{
                var xs,dy:Array;
                var pt:Point;
                var eps:Number=0.1;
                var y1,y2:Number;
                var ints:Boolean=false;          //相交标志
                xs=[L,0,R];      //左、中、右的位置
                dy=new Array(3);     //x 在左、中、右位置时两
//曲线的 y 值之差
                while(xs[0]<xs[2]){
                    if(xs[2]-xs[0]<eps) break; //子区间长度
//很小时,终止循环
                    xs[1]=(xs[0]+xs[2])/2;       //在[L,R]中插
//入中点 M
                    for(var i:int=0;i<3;i++){
                        y1=ca.getYByX(xs[i])[0];
                        y2=cb.getYByX(xs[i])[0];
                        dy[i]=y1-y2;
                        if(Math.abs(dy[i])<eps){
                            pt=new Point(xs[i],(y1+y2)/2);
                            ints=true;
                            break;
                        }
                    }
                    if(ints) break;
                    if(dy[0]*dy[1]<0){dy[2]=dy[1];continue}
    //左段中包含交点
                    if(dy[1]*dy[2]<0){dy[0]=dy[1];continue}
    //右段中包含交点
                }
                return pt;
        }
```

2.2.2 存在问题

与前面的扫描法相比，效率明显提高，但缺陷仍然不可避免：

(1) 局限于只有一个交点的情形。若有多个交点，仅依靠上述代码无法求全。如图 4 (a)，左、右两子区间，两端 y 值之差均异号，代码 3 中最后两个判断句均为真，但实际执行时，当执行了第一个判断，取了左子区间后，遇到 continue 语句，就开始对左子区间继续进行两分查找，而第二个判断则执行不到，从而把右子区间给舍去了。因而，只能求出左段包含的交点。而如果将 continue 语句去掉，则进入下一循环时，L=R=M，不满足循环条件，结束循环，于是一个交点也求不到。

(2) 在根据 x 值求 y 值时仍然面临着无值、多值问题。如图 4 (b)，当 x=L 时，求曲线 A 的 Y 值时有两个，求曲线 B 的 Y 值时，无值，从而无法求得 ldy；当 x=R 时也有类似情况而无法求得 rdy。

(3) 上下相切时 (如图 4 (c))，ldy，mdy，rdy 始终同号，切点将无法求得。

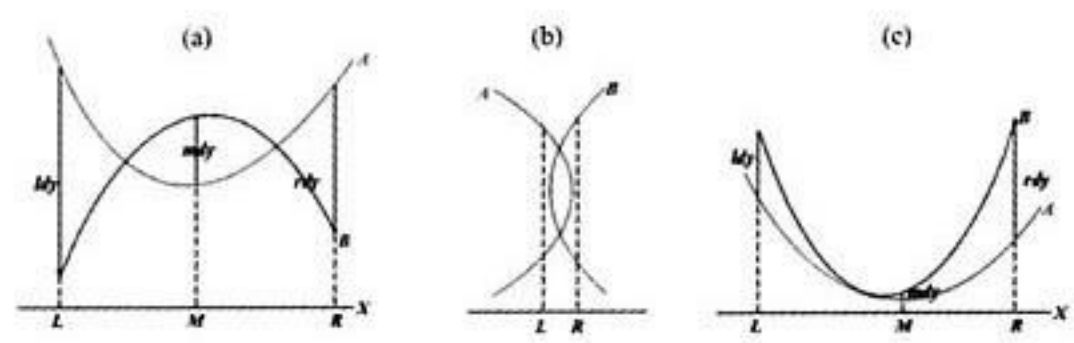


图 4　两分查找法求交点存在问题

2.2.3 改进方法

对于问题(1)，改进办法是将两分查找法与分治法相结合。先将两曲线相交区域的 x 值域 [L,R] 分成若干区间，对每一区间运用两分查找法。但随之而来的问题是对区间 [L,R] 如何划分。这里采用的方法是，将相交区域的左、右端及两曲线所有位于相交区域内的控制点的横坐标值放入一个数组，将数组升序排列。这样数组中每相邻两元素便构成一个区间，再对每一区间运用两分查找法求出每一段中包含的交点。

对于问题②，类似于 1.1.3 中对相似问题的改进，在横向分割存在无值、多值问题时，改用纵向分割。即对相交区域的 y 值域进行分割，这样就变成以 x 值求 y 值为根据 y 值求 x 值，原来的问题在大部分情况下，可以得到解决，但仍有例外。

问题(3)在两分查找法中无法解决。

2.3　牛顿法

2.3.1 算法与代码

这也是受《数学手册》中牛顿法解代数方程的启示。本来的思路是：从曲线的一个端点出发画切线，与 x 轴交于 x_1，再画曲线在 x_1 处的切线交 x 轴交于 x_2，……直到切点与 x 轴的距离小于给定的精度 ε 为止。

该算法应用于曲线求交时，步骤如下：

①过两曲线的起点分别画切线 L_A、L_B (在 Bezier 曲线中就是前两个控制点的连线)；如图 5 所示。

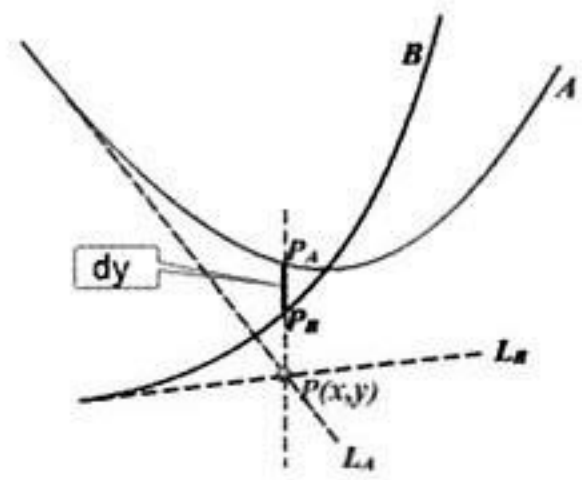


图 5　牛顿法求交点算法描述

②求出两切线的交点 (记为 P (x,y))。

③过 P 画垂线交曲线 A、B 于 P_A、P_B。

④计算 P_A、P_B 的距离 dy，如果 dy<ε，则将点 (x, (PA.y+PB.y) /2) 作为交点的近似。否则，过 P_A 作曲线 A 的切线，过 P_B 作曲线 B 的切线，重复②、③、④。

这里事实上隐含了数学中常用的化曲为直思想，将求曲线交点转化为求直线交点。AS3.0 代码如下：

```
//代码 4  Newton 法求两曲线交点
        //参数 ca、cb 表示两曲线
        //参数 la、lb 为两曲线过起点的切线
        //返回两曲线的交点
function newton(ca,cb,la,lb):Point {
                var xv,ya,yb,dy:Number;
                var p,pa,pb,pt:Point;
                var eps:Number=0.1;
                while (true) {
                        p=la.intersection(lb);
                        xv=p.x;
                        ya=ca.getYByX(xv)[0];
                        yb=cb.getYByX(xv)[0];
                        dy=Math.abs(ya-yb);
                        if (dy<eps) {
                                pt=new Point(xv,(ya+yb)/2);
                                break;
                        }
                        pa=new Point(xv,ya);
                        pb=new Point(xv,yb);
        la=ca.getTan(pa);     //求曲线 A 过点 pa 的切线
        lb=cb.getTan(pb);     //求曲线 B 过点 pb 的切线
                }
                return pt;
        }
```

上述代码中根据曲线上一点求过该点切线的函数 getTan (P)，需先求出曲线在该点的斜率，再根据给定点、给定的切线长度及斜率求出切线段另一端点的坐标，即可得到切线。

2.3.2 存在问题

对于正常情况，循环二、三步即可得到一个交点，效率比两分查找法又有提高，但 2.2.2 节中的前两个问题还是存在。

2.3.3 改进方法

要解决求多个交点的问题，总体思路还是分而治之的分治法，但分的方法与两分查找法不同。这里用的方法是，选阶数较低曲线两端点的连线作为 L_A，另一曲线的控制点两两相连所成直线为 L_{B1}，L_{B2}……，再分别对 L_A 与 L_{B1}，L_A 与 L_{B2}，……使用牛顿法求交 (即重复本算法描述中第②、③、④步。假定曲线 ca、cb 的次数为 na、nb，控制点数组为 pa、pb，且 na<nb，则可用以下代码实现分治法：

```
//代码 5,分治法与牛顿法结合求交点
la.byPP(pa[0],pa[na]);
for(var i:int=0;i<nb;i++){
        lb.byPP(pb[i],pb[i+1]);
        pt=newton(ca,cb,la,lb);     //调用牛顿法求交点函数
        if(pt! =null) pts.push(pt);
}
```

代码中函数 byPP (p1,p2) 为根据两端点重新定义直线段。

无值、多值问题的解决。鉴于 y 值的个数取决于 t 值的个数，因此把由 x 值求 y 值转变为由 x 值求 t 值，得到 t 值组成的数组：tsx=crv.getTByX (P.x)。为了避免过 P 点的垂线与曲线不相交，从而 tsx 数组为空，再根据该点的纵坐标求一下 t 值，得到另一个 t 值组成的数组 tsy=crv.getTByY (P.y)，两个数组元素的个数在 0-n 之间，既可能无值，也可能多值。而只需要一个值，到底取哪一个呢？约定一个原则：所取 t 值对应的曲线上点与给定点 P 的距离是所有 t 值中最小的。为此，要做以下 3 步：

①在数组 tsx 中找出与给定点 P 距离最近的 t 值 tx 及最小距离 dx。

若数组 tsx 非空，对 tsx 中的每一个 t 值，求出相应的 y 值，及与给定点 P 的纵坐标的距离，取距离最小时的 t 值为 tx，最小距离记为 dx。

若数组 tsx 为空，用给定点 P 与曲线两端点的距离大小进行取舍，若 P 与起点较近，取 t=0,dx 为 P 与起点的距离；否则若 P 与终点较近，则取 t=1,dx 为 P 与终点的距离。

②在数组 tsy 中找出与给定点 P 距离最近的 t 值 ty 及最小距离 dy。

若数组 tsy 非空，对 tsy 中的每一个 t 值，求出相应的 x 值，及与给定点 P 的横坐标的距离，取距离最小时的 t 值为 ty，最小距离记为 dy。

若数组 tsy 为空，用与 tsx 为空同样的方法取 t=0 或 t=1。

③比较 dx 与 dy，若 dx≤dy，取 tx，否则取 ty。

据此，可得到代码：

```
//代码 6,根据给定点(不一定在曲线上)求曲线的唯一参数值
//参数 crv 为曲线,p 为给定点
//返回曲线的唯一参数值
function getTByP(crv:McCurve,p:Point):Number{
                var x,y,tx,ty,dx,dy,d0,d1:Number;
                var tsx,tsy:Array;
                var i:int;
                tsx=crv.getTByX(p.x);
                tsy=crv.getTByY(p.y);
                if(tsx.length<1){
                        d0=Point.distance(p,crv.ps[0]);
                        d1=Point.distance(p,crv.ps[crv.n]);
                        if(d0<d1){ tx=0;dx=d0}
                        else{tx=1;dx=d1};
                }else{
                        dx=100;
```

```
        for(i=0;i<tsx.length;i++){
            d0=Math.abs(crv.getY(tsx[i]),p.y);
            if(d0<dx){d0=dx;tx=tsx[i];}
        }
    }
    if(tsy.length<1){
        d0=Point.distance(p,crv.ps[0]);
        d1=Point.distance(p,crv.ps[crv.n]);
        if(d0<d1){ ty=0;dy=d0}
        else{ty=1;dy=d1};
    }else{
            dy=100;
            for(i=0;i<tsy.length;i++){
                d0=Math.abs(crv.getX(tsy[i]),p.x);
                if(d0<dy){d0=dy;ty=tsy[i];}
            }
        }
        if(dx<dy) return tx;
        else return ty;
    }
```

这样便从根本上解决了无值、多值问题，对于 la 与 lb 的交点 P，总可以得到曲线 ca、cb 的唯一一个参数值 ta、tb，从而可以求出曲线上与参数对应的点 pa，pb 和斜率 ka、kb，也可求出 pa、pb 两点的距离，过 pa、pb 的曲线的切线，从而可以把代码 4 改写成

代码 7，牛顿法求交点代码的改进

```
function newton(ca,cb,la,lb):Point{
        var p,ret:Point;
        var d:Number;
        var ta,tb:Number;     //两曲线的参数值
        var xa,ya,xb,yb:Number;   //与参数值对应的
//两曲线上点的坐标
        var ka,kb:Number;    //曲线的斜率
        var count:int=0;
        while(count<5){
            p=la.intersection(lb);
            if(p==null) break;
            ta=ca.getTByP(p);
            xa=ca.getX(ta);
            ya=ca.getY(ta);
            ka=ca.getK(ta,"t");
            pa=new Point(xa,ya);
            tb=cb.getTByP(p);
            xb=cb.getX(tb);
            yb=cb.getY(tb);
            kb=cb.getK(tb,"t");
            pb=new Point(xb,yb);
            d=Point.distance(pa,pb);
            if(d<1){
            return new Point((xa+xb)/2,(ya+yb)/2);
            }
            la.byPK(pa,ka);
            lb.byPK(pb,kb);
            count++;
        }
        return ret;
    }
```

2.4 离散法

2.4.1 算法描述

这是王国瑾等给出的 Bezier 曲线求交方法，其基本思想还是化曲为直，将曲线求交转化成直线求交。但转化的方法与牛顿法不同，这里采用的是分割化直的办法，即将曲线分割成若干等次的 Bezier 曲线，当分割得很小时，用子曲线段两端点的连线代替子曲线段，然后求一曲线的所有子曲线替代直线段与另一曲线的所有子曲线替代直线段的交点，求得的交点即为原曲线交点的近似解。其堆栈算法描述如下：

(1) 初始化：

计算两曲线的先验分割次数(决定分割到何时为止)ar0,br0;

计算两曲线的包围盒

如果两包围盒不相交,则返回一个空数组,否则

将两曲线的控制点数组、当前分割次数 (=0) 压入堆栈。

(2) 循环 (直到堆栈为空止)

{

从堆栈中弹出两子曲线的控制点数组(记为 aps,bps)，两曲线的当前分割次数(ar,br)

比较 ar 与 ar0，br 与 br0 的大小，根据比较结果分别处理{

情况 1：ar≥ar0,br≥br0(两曲线的分割都已经够小了)

连结子曲线 A 的两个端点，得到直线段 L_A

连结子曲线 B 的两个端点，得到直线段 L_B

求 L_A 与 L_B 的交点

如果交点存在，将交点存入结果数组

情况 2：ar<ar0, br≥br0(子曲线 A 尚待继续分割，子曲线 B 已分割到位)

将子曲线 A 分割成两段 A_1，A_2，并将 ar+1

判断 A_1 与 B 的包围盒是否相交，如相交，将 A_1、B 的控制点数组、各自的当前分割次数压入堆栈

对 A_2 重复上述操作

情况 3：ar≥ar0,br<br0(子曲线 A 分割到位，子曲线 B 尚待继续分割)

将子曲线 B 分割成两段 B_1，B_2，并将 br+1

判断 A 与 B1 的包围盒是否相交，如相交，将 A、B_1 的控制点数组、各自的当前分割次数压入堆栈

对 B_2 重复上述操作

情况 4：ar<ar0,br<br0(两子曲线都需继续分割)

将子曲线 A 分成 A_1，A_2 两段，ar+1

将子曲线 B 分成 B_1，B_2 两段，br+1

分别检查 A_1 与 B_1、B_2，A_2 与 B_1、B_2 的包围盒是否相交，若相交，将相应子曲线段的控制点数组及其当前分割次数压入堆栈

}

}

上述算法已在王国瑾给出的算法上作了些许改进，改进之处为包围盒相交性检查的位置。原算法在曲线分割后，不管子曲线与另一曲线 (或其子曲线) 的包围盒是否相交，不作判断，直接压入堆栈，在循环开始从堆栈弹出后再行检查。这样，无疑增加了大量的堆栈压入、弹出操作，浪费了机时；同时也会使堆栈的空间占用大量增加。据此，我们将包围盒相交性检查放在曲线分割后，压入堆栈之前，对于包围盒不相交的子曲线段直接无视，不再压入堆栈，仅把包围盒相交的子曲线段压入堆栈。虽然在代码上增加了不少，但实际检查次数并未增加，且在减少堆栈空间占用的同时，减少了大量无效的堆栈压入、弹出操作。实验证明，速度明显加快。

2.4.2 算法实现

上述算法，在实现时还有一些具体工作要做：

(1) 先验分割次数的计算

王国瑾给出了先验分割次数 (先验离散层数) 的计算公式：

$r_0 = \log_2[(n/(n-1)L_0)/(8\varepsilon)]/2$

其中：n 为 Bezier 曲线的次数。

ε 为给定的精度 (一个小正数)，使分割后子曲线上任一点到子曲线段两端点连线的距离小于此值。

$L_0 = \max_{0\le i\le n-2}\|\Delta^2 P_i\|$，$\Delta P_i = P_{i+1} - P_i$，$P_i$为曲线的第 i 个控制点。

实际上，为了简便，可视曲线的弯曲程度直接给定一个先验分割次数。

(2) 曲线包围盒相交性判断

如果运用自定义的继承于 MovieClip 类的 McCurve 类，可用 MovieClip 类的 getBounds ()，getRect () 方法获得曲线的包围盒。但鉴于本算法用到的仅是曲线的控制点，而 McCurve 类包含了大量与本算法无关的属性，建立众多该类的实例，难免会增加空间占用。因此在该算法实现上，我们没有用 McCurve 类。于是便不能直接应用上述获取包围盒的方法，需另行定义获取方法，当然这并不困难，无非是求出所有控制点的纵、横坐标的最小值、最大值，据此构造一个 Rectangle 类实例。进而可用 Rectangle 类的 intersects 方法判断两矩形是否相交。

(3) Bezier 曲线的分割

这在不少资料上有介绍，可在 t1=0.5 处执行 de Casteljan 递推算法。

$P_i^k = \begin{cases} P_i & k=0 \\ (1-t)P_i^{k-1} + tP_{i+1}^{k-1} & k=1\cdots n;i=0\cdots n-k \end{cases}$，其中

P_i (i=0……n) 是 n 阶 Bezier 曲线的 n+1 控制点，t ∈ [0,1]。所得的两条子 Bezier 曲线的控制点分别为：

P_0^i (i=0……n) ，t ∈ [0,t1] 和 P_j^{n-j} (j=0,……n) ，t ∈ [t1,1]。

代码 8 Bezier 曲线分割的 de Casteljan 递推算法

```
//参数 pts 为曲线的控制点数组,t 为分割点的参数值
//返回两子曲线段的控制点数组
function seg(pts:Array,t:Number=0.5):Array {
                var n:int=pts.length-1;
                var P:Array=new Array(n);
                var sub1,sub2:Array;
                var i,j,k:int;
                sub1=new Array(n);
                sub2=new Array(n);
                k=0;
                P[k]=new Array(n);
                for (i=0; i<=n; i++) {
                    P[k][i]=pts[i];
                }
                for (k=1; k<=n; k++) {      //de Casteljan 递推
                    P[k]=new Array(n-k);
                    for (i=0; i<=n-k; i++) {
                        P[k][i]=new Point();
P[k][i].x=(1-t)*P[k-1][i].x+t*P[k-1][i+1].x;
P[k][i].y=(1-t)*P[k-1][i].y+t*P[k-1][i+1].y;
                    }
                }
                for (k=0; k<=n; k++) {
//左边子曲线段的控制点数组
                    sub1[k]=P[k][0];
                }
                for (k=n; k>=0; k--) {
//右边子曲线段的控制点数组
                    sub2[n-k]=P[k][n-k];
                }
                var bs:Array=new Array(2);
                bs[0]=sub1;
                bs[1]=sub2;
                return bs;
    }
```

2.4.3 存在问题与改进办法

该算法求交速度快，适用性好，而且避免了 getYByX () 或 getXByY () 可能出现的无值、多值问题。但在两曲线相切或接近相切时，也会出现交点个数虚增的情况。对此的改进办法是，在算法情况 1 中，求得一个非空交点后，将堆栈中所有分割次数大于曲线次数-1 的元素全部弹出，不作处理，这样既节约了时间，又避免了交点个数的虚增。

另外一个需要注意的问题是，在求连线交点时，所得交点可能在线段之外，这可能导致所求交点不在两曲线之上。改进办法是判断所求交点是否位于两连线段的包围盒相交区域内，若是则返回该交点，否则返回 null。在 AS3.0 中可以借助Rectangle 类的 intersection 方法求得两连线段包围盒的相交区域，再用其 containPoint (point) 判断点是否在相交区域内。

2.5 代数法 (解非线性方程组法)

若曲线由显函数表示，如曲线 A：y=f (x)，曲线 B：y=g

(x)，则求曲线 A、B 的交点只需解方程组 $\begin{cases} y=f(x) \\ y=g(x) \end{cases}$ 即可。但 Bezier 曲线通常只能用参数方程表示，现记曲线 A 的参数方程为：$\begin{cases} x=\sum_{i=0}^{na} a_i s^i \\ y=\sum_{i=0}^{na} b_i s^i \end{cases}$，曲线 B 的参数方程为 $\begin{cases} x=\sum_{i=0}^{nb} a'_i t^i \\ y=\sum_{i=0}^{nb} b'_i t^i \end{cases}$，s，t∈ [0,1]

分别为曲线 A、B 的参数，na，nb 分别为曲线 A、B 的次数，a_i、b_i、a_i'、b_i' 为参数方程的系数。于是要求曲线 A、B 的交点，则需解以下两元 Max (na,nb) 次方程组：

$$\begin{cases} \sum_{i=0}^{na} a_i s^i - \sum_{i=0}^{nb} a'_i t^i = 0 \cdots\cdots (1) \\ \sum_{i=0}^{na} b_i s^i - \sum_{i=0}^{nb} b'_i t^i = 0 \cdots\cdots (2) \end{cases}$$。为方便将方程组记为 $\begin{cases} u(s,t)=0 \\ v(s,t)=0 \end{cases}$

当曲线的阶数≥2 时，传统的消元法，矩阵法便无能为力，因而常到此却步不前。偶而在《数学手册》中发现“牛顿法解非线性方程组”一节，顿时感觉眼睛一亮，真是踏破铁鞋无觅处，得来全不费功夫。

2.5.1 算法描述

从一组近似解 P_0 (s_0，t_0) 开始用迭代公式：

$$s_{n+1}=s_n+\frac{1}{J_n}\begin{vmatrix} \frac{\partial u}{\partial t} & u \\ \frac{\partial v}{\partial t} & v \end{vmatrix}_{P_n},\quad t_{n+1}=t_n+\frac{1}{J_n}\begin{vmatrix} u & \frac{\partial u}{\partial s} \\ v & \frac{\partial v}{\partial s} \end{vmatrix}_{P_n}$$，其中

$$J_n=\begin{vmatrix} \frac{\partial u}{\partial s} & \frac{\partial u}{\partial t} \\ \frac{\partial v}{\partial s} & \frac{\partial v}{\partial t} \end{vmatrix}_{P_n}$$

直到据此求得的两曲线上的点 PAn+1 与 PBn+1 之间的距离小于给定的精度 ε 为止。

2.5.2 算法实现

作为迭代出发点的近似解的确定，可以用牛顿法中的方法，取次数较低曲线两端点的连线作为 L_A，取另一曲线的控制点两两相连所成直线为 L_{B1}、L_{B2}……，再求 L_A 与 L_{B1}、L_{B2}……的交点，根据每一个交点用 1.3.3 中的 getTByP (crv,p) 求出两曲线的参数 s0，t0，作为方程组的近似解开始迭代。

各偏导数的计算。由于 u，v 均为多项式函数，故求偏导数并不困难，可用以下公式直接求得：

$$\frac{\partial u}{\partial s}=\sum_{i=1}^{na} i\cdot a_i\cdot s^{i-1},\quad \frac{\partial u}{\partial t}=-\sum_{i=1}^{nb} i\cdot a'_i\cdot t^{i-1}$$

$$\frac{\partial v}{\partial s}=\sum_{i=1}^{na} i\cdot b_i\cdot s^{i-1},\quad \frac{\partial v}{\partial t}=-\sum_{i=1}^{nb} i\cdot b'_i\cdot t^{i-1}$$。

行列式的计算，由于迭代过程中用到的都是两阶行列式，因此只要对角相乘再相减即可。

实验表明，该算法效率最高，准确性最高，但也会有交点数虚增的现象，但造成虚增的原因不在于算法的核心部分，而是由于分治法，在不该分治时分治所致。

3 各种求交算法比较

对各种求交算法针对 7 组不同的曲线分别求交 40 次，计算每一次求交的运行时间，实验结果如下 (实验环境：Intel Pentium 4，2.93GHz CPU,736M 内存，Microsoft WindowsXP SP2,Adobe Flash Player10)：

时间(ms) 方法 / 曲线组	扫描法	两分查找法	牛顿法	离散法	代数法
	5.25	1.475	1.675	1.175	0.425
	5.075	2.125	1.275	1.525	0.4
	6.525	2.6	1.15	1.55	0.55
	10.45	4.2	1.8	4.2	0.625
	28.65	6.375	2.475	2.3	1.15
	11.025	4.15	1.75	3.85	0.65
	22.625	3.6	1.775	3.7	0.775
平均	12.800	3.504	1.700	2.614	0.654
最大	28.650	6.375	2.475	4.200	1.150
最小	5.075	1.475	1.150	1.175	0.400
极差	23.575	4.900	1.325	3.025	0.750
标准差	8.556	1.511	0.397	1.178	0.236

可见，诸方法中以解非线性方程组的方法效率最高，也最稳定。平均求交时间仅 0.654 毫秒，标准差仅 0.236 毫秒，极差也只有 0.75 毫秒，说明面对各种不同情况求交时间差异不会太大。而扫描法是效率最低的方法，即便用了改变步长的方法，其平均求交时间仍然高达 12.8 毫秒，而且稳定性差，求交时间的极差高达 23.575 毫秒，标准差也达 8.556 毫秒。

参考文献

[1] 《数学手册》编写组. 数学手册 [M] . 北京：高等教育出版社，1979，5.

[2] 曹锋. 基于 Bezier 曲线求交的曲线裁剪算法 [J] . 计算机应用，1998，8：20-22.

[3] 王国瑾，等. 计算机辅助几何设计 [M] . 高等教育出版社、施普林格出版社，2001.

[4] 李原，张开富，余剑峰. 计算机辅助几何设计技术及应用 [M] . 西北工业大学出版社，2007.

（收稿日期：2012-08-16）

在 C++ Builder 中定制 Excel 2003 图表

杜希国

摘　要：Microsoft Excel 2003 是一款实用性与功能性很强的表格处理软件，可以方便快捷地进行数据处理，而其图表功能，可直观地反映数据情况。结合 C++ Builder 程序，利用 OLE 技术，可以更好地将数据计算与表格处理相结合，增强了数据处理的灵活性，提高了工作效率。

关键词：C++ Builder 语言；OLE 技术；图表

1　引言

Microsoft Excel 2003 是 Microsoft Office 系列产品中一款专门用于表格处理的软件，功能强大，使用方便，应用广泛，在各类报表处理中都可见到它的身影。图表作为一种形象直观的数据表现形式，也得到了普遍使用。

OLE，即对象链接与嵌入，是应用程序间交换数据、相互操作的一种有效方式，Office 系列产品提供了很强的 OLE 服务功能。C++ Builder 采用间接的办法，利用变体类 Variant 所提供的 4 个方法向 OLE 服务程序提交操纵命令，可以实现在 C++ Builder 中操纵 Excel，充分将程序设计的灵活性与 Excel 表格处理的高效性结合起来，提高报表设计与处理的效率。

接下来就如何在 C++ Builder 中定制 Excel 2003 图表的相关内容进行详细介绍。

2　基础知识

首先认识 Excel 2003 中的图表。

定义 3 个变量：Variant Sheet，Chart，Range; Sheet 为工作表对象，Chart 为图表对象，Range 为数据区域对象。

启动 Excel 2003 应用程序，点击“插入”菜单下的“图表”选项，弹出“图表向导”对话框。

2.1　图表类型

图 1　图表类型

图表类型分为标准类型和自定义类型，如图 1 所示。在此着重介绍标准类型的相关知识。标准类型共有 14 类，每类又分为不同样式。选择图表类型的语法形式为：

```
Chart.OlePropertySet("ChartType",ChartTypeValue);
```

说明：ChartTypeValue 的取值可以如表 1–表 14 中的常量值，也可直接设置为数值。

表 1　柱形图类型及取值

常量	取值	类型
xlColumnClustered	51	簇状柱形图
xlColumnStacked	52	堆积柱形图
xlColumnStacked100	53	百分比堆积柱形图
xl3DColumnClustered	54	三维簇状柱形图
xl3DColumnStacked	55	三维堆积柱形图
xl3DColumnStacked100	56	三维百分比堆积柱形图
xl3DColumn	–4100	三维柱形图

表 2　条形图类型及取值

常量	取值	类型
xlBarClustered	57	簇状条形图
xlBarStacked	58	堆积条形图
xlBarStacked100	59	百分比堆积条形图
xl3DBarClustered	60	三维簇状条形图
xl3DBarStacked	61	三维堆积条形图
xl3DBarStacked100	62	三维百分比堆积条形图

表 3　折线图类型及取值

常量	取值	类型
xlLine	4	折线图
xlLineStacked	63	堆积折线图
xlLineStacked100	64	百分比堆积折线图
xlLineMarkers	65	数据点折线图
xlLineMarkersStacked	66	堆积数据点折线图
xlLineMarkersStacked100	67	百分比堆积数据点折线图
xl3DLine	-4101	三维折线图

表 4　饼图类型及取值

常量	取值	类型
xlPie	5	饼图
xl3DPie	-4102	三维饼图
xlPieOfPie	68	复合饼图
xlPieExploded	69	分离型饼图
xl3DPieExploded	70	分离型三维饼图
xlBarOfPie	71	复合条饼图

表 5　XY 散点图类型及取值

常量	取值	类型
xlXYScatter	-4169	散点图
xlXYScatterSmooth	72	平滑线散点图
xlXYScatterSmoothNoMarkers	73	无数据点平滑线散点图
xlXYScatterLines	74	折线散点图
xlXYScatterLinesNoMarkers	75	无数据点折线散点图

表 6　面积图类型及取值

常量	取值	类型
xlArea	1	面积图
xlAreaStacked	76	堆积面积图
xlAreaStacked100	77	百分比堆积面积图
xl3DArea	-4098	三维面积图
xl3DAreaStacked	78	三维堆积面积图
xl3DAreaStacked100	79	三维百分比堆积面积图

表 7　圆环图类型及取值

常量	取值	类型
xlDoughnut	-4120	圆环图
xlDoughnutExploded	80	分离型圆环图

表 8　雷达图类型及取值

常量	取值	类型
xlRadar	-4151	雷达图
xlRadarMarkers	81	数据点雷达图
xlRadarFilled	82	填充雷达图

表 9　曲面图类型及取值

常量	取值	类型
xlSurface	83	三维曲面图
xlSurfaceWireframe	84	三维曲面图（框架图）
xlSurfaceTopView	85	曲面图（俯视图）
xlSurfaceTopViewWireframe	86	曲面图（俯视框架图）

表 10　气泡图类型及取值

常量	取值	类型
xlBubble	15	气泡图
xlBubble3DEffect	87	三维气泡图

表 11　股价图类型及取值

常量	取值	类型
xlStockHLC	88	盘高-盘低-收盘图
xlStockOHLC	89	开盘-盘高-盘低-收盘图
xlStockVHLC	90	成交量-盘高-盘低-收盘图
xlStockVOHLC	91	成交量-开盘-盘高-盘低-收盘图

表 12　圆柱图类型及取值

常量	取值	类型
xlCylinderColClustered	92	柱形圆锥图
xlCylinderColStacked	93	堆积柱形圆锥图
xlCylinderColStacked100	94	百分比堆积柱形圆柱图
xlCylinderBarClustered	95	条形圆柱图

常量	取值	类型
xlCylinderBarStacked	96	堆积条形圆柱图
xlCylinderBarStacked100	97	百分比堆积条形圆柱图
xlCylinderCol	98	三维柱形圆柱图

表 13 圆锥图类型及取值

常量	取值	类型
xlConeColClustered	99	柱形圆锥图
xlConeColStacked	100	堆积柱形圆锥图
xlConeColStacked100	101	百分比堆积柱形圆锥图
xlConeBarClustered	102	条形圆锥图
xlConeBarStacked	103	堆积条形圆锥图
xlConeBarStacked100	104	百分比堆积条形圆锥图
xlConeCol	105	三维柱形圆锥图

表 14 棱锥图类型及取值

常量	取值	类型
xlPyramidColClustered	106	柱形棱锥图
xlPyramidColStacked	107	堆积柱形棱锥图
xlPyramidColStacked100	108	百分比堆积柱形棱锥图
xlPyramidBarClustered	109	条形棱锥图
xlPyramidBarStacked	110	堆积条形棱锥图
xlPyramidBarStacked100	111	百分比堆积条形棱锥图
xlPyramidCol	112	三维柱形棱锥图

2.2 图表源数据

图表源数据设置如图 2 所示。

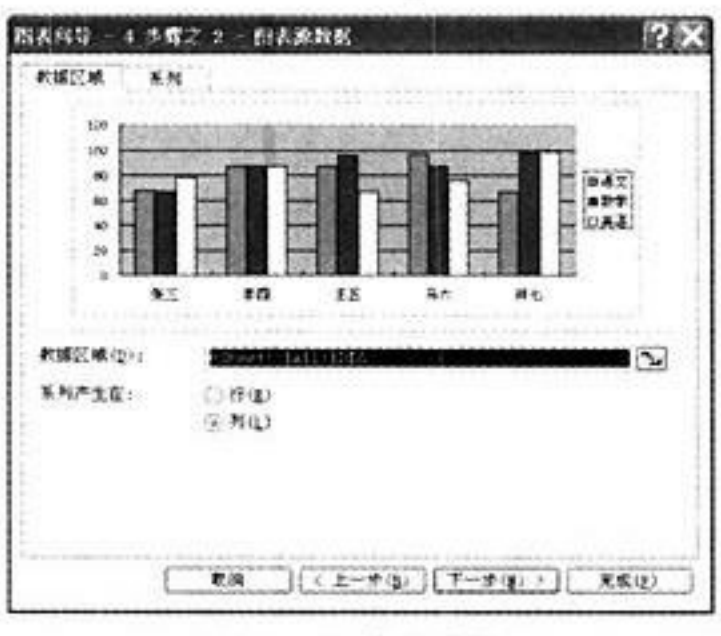


图 2 图表源数据

2.2.1 设置图表的数据区域

```
Range=Sheet.OlePropertyGet("Range","A2:D8"); //选择
//A2:D8 区域生成图表
```

2.2.2 设置图表的系列

xlRows 表示数据是以列来做排列，xlColumns 表示用行来做排列。如果是 xlRows，选择范围的第一列就是 X 轴数据的名称，第一行是 Y 轴数据的名称，其余的作为图表里的数值。

```
Chart.OleProcedure("SetSourceData",Range,xlRows);
//设置图表以行做排列
```

2.3 图表选项

2.3.1 标题

图表选项的标题设置如图 3 所示。

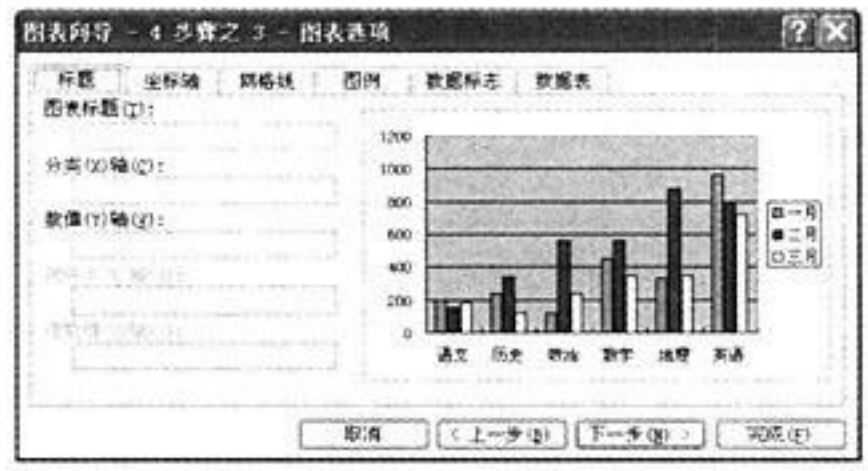


图 3 图表选项之标题

（1）图表标题

```
Chart.OlePropertySet("HasTitle",true);
AnsiString Charttitle="书籍销售情况图"
    Chart.OlePropertyGet    ("ChartTitle").OlePropertySet
("Text",Charttitle.c_str());
```

（2）坐标轴标题

一般情况下，图表有两个用于对数据进行分类和度量的坐标轴，即分类（X）轴和数值（Y）轴，三维图表有第三个（Z）轴，而饼图和圆环图则没有坐标轴。坐标轴类型及取值如表 15 所示。

表 15 坐标轴类型及取值

常量	取值	类型
xlCategory	1	分类
xlValue	2	数值
xlSeries	3	系列

若设置坐标轴标题，则需先设置该坐标轴的“HasTitle”属性为 true，然后利用“AxisTitle”属性设置坐标轴标题内容。

以设置分类轴标题为例：

```
Chart.OlePropertyGet("Axes",xlCategory).OlePropertySet
("HasTitle",true);
                Chart.OlepropertyGet  ("Axes",xlCategory).
```

OlePropertyGet("AxisTitle").OlePropertySet("Text","名称");

2.3.2 坐标轴

图表选项的坐标轴设置如图 4 所示。

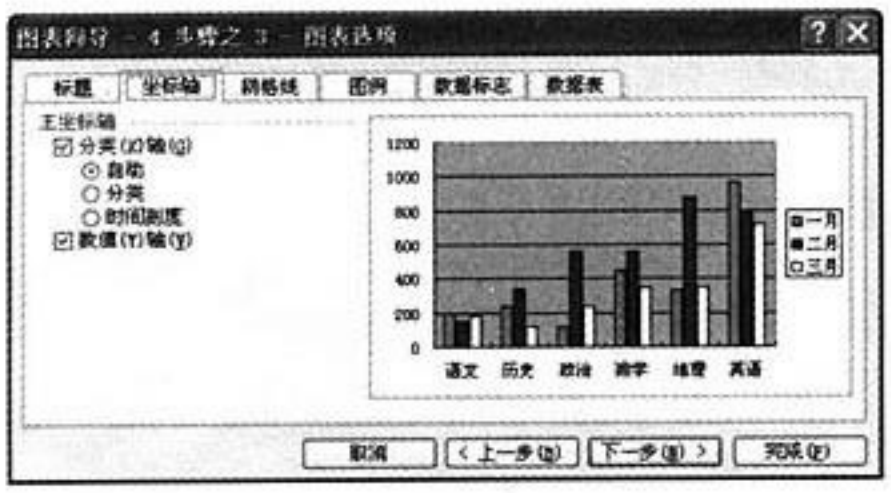


图 4　图表选项之坐标轴

(1) 坐标轴显示

在生成的图表中，可以通过“HasAxis”属性进行选择是否显示坐标轴，以及显示哪个坐标轴。

以分类轴为例，其语法形式为：

Chart.OlePropertySet("HasAxis",xlCategory,false); //不显示分类轴

(2) 分类轴设置

分类（X）轴有 3 种不同形式，取值及形式如表 16 所示。

表 16　分类坐标轴类型及取值

常量	取值	类型
xlCategoryScale	2	分类
xlTimeScale	3	时间刻度
xlAutomaticScale	–4105	自动

其语法形式为：

Chart.OlePropertyGet ("Axes",xlCategory).OlePropertySet ("CategoryType",xlTimeScale)

2.3.3 网格线

图表选项的网格线设置如图 5 所示。

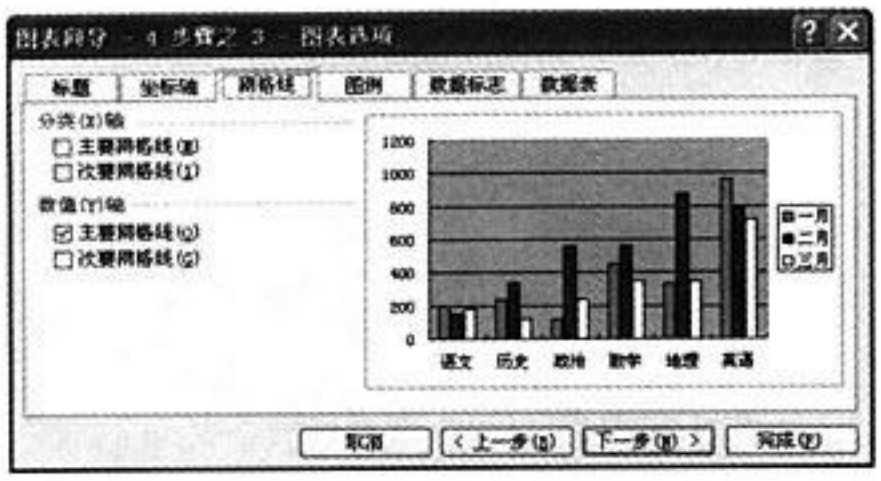


图 5　图表选项之网格线

通过“HasMajorGridLines”属性和“HasMinorGridLines”属性，可以分别设置是否显示坐标轴的主要网格线和次要网格线。

分类轴显示主要网格线和次要网格线的语法为：

Chart.OlePropertyGet ("Axes",xlCategory).OlePropertySet ("HasMajorGridLines",true);

Chart.OlePropertyGet ("Axes",xlCategory).OlePropertySet ("HasMinorGridLines",true);

数值轴显示主要网格线和次要网格线的语法为：

Chart.OlePropertyGet ("Axes",xlValue).OlePropertySet ("HasMajorGridLines",true);

Chart.OlePropertyGet ("Axes",xlValue).OlePropertySet ("HasMinorGridLines",true);

2.3.4 图例

图表选项的图例设置如图 6 所示。

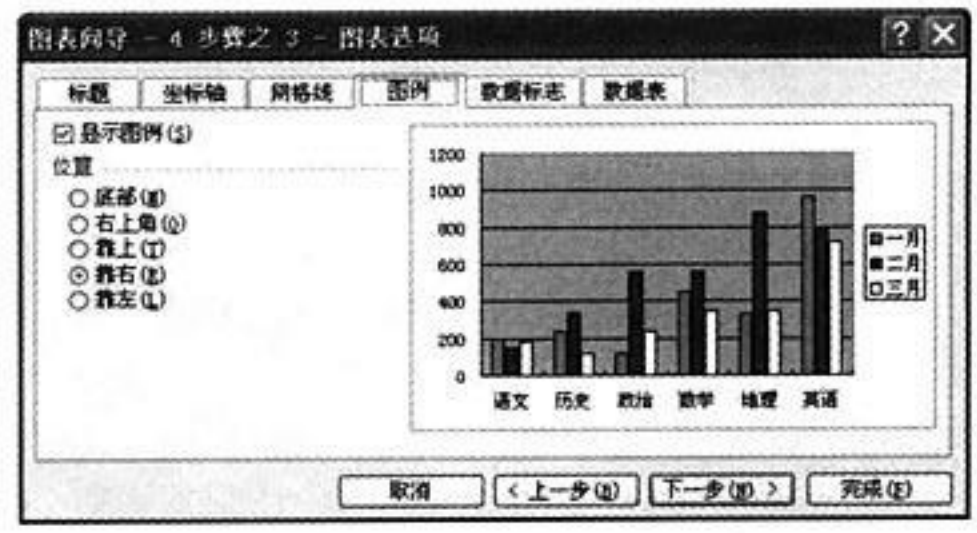


图 6　图表选项之图例

图例是一个方框，用于标识为图表中的数据系列或分类指定的图案或颜色。

(1) 显示图例

通过“HasLegend”属性设置是否显示图例，该属性设置 true 为显示图例，false 则不显示。

Chart.OlePropertySet("HasLegend",true);

(2) 位置

图例的显示位置，其取值与位置对应关系如表 17 所示。

表 17　图例位置及取值

常量	取值	类型
xlLegendPositionBottom	1 或–4107	底部
xlLegendPositionCorner	2	右上角
xlLegendPositionTop	3 或–4160	靠上
xlLegendPositionRight	4 或–4152	靠右
xlLegendPositionLeft	5 或–4131	靠左

其语法形式为：

Chart.OlePropertyGet ("Legend").OlePropertySet ("Position",–4160);

2.3.5 数据标志

图表选项的数据标志设置如图 7 所示。

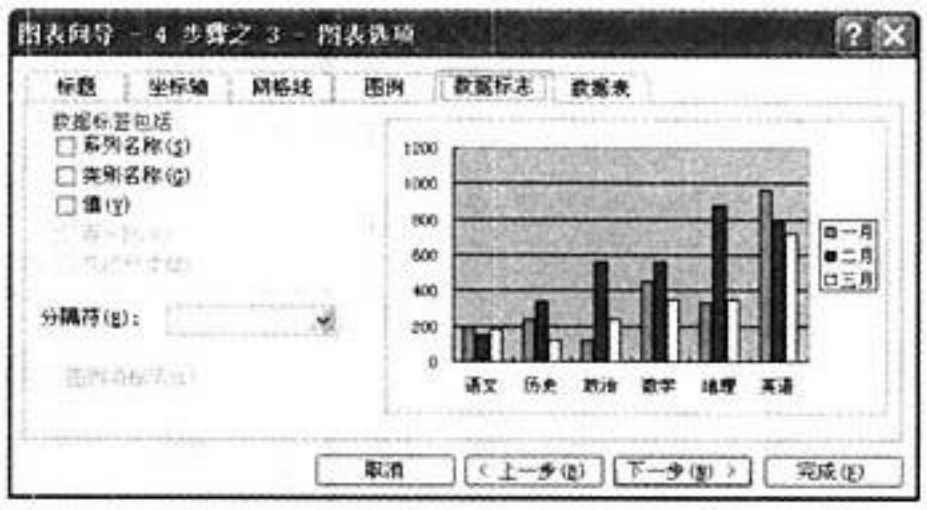


图 7 图表选项之数据标志

(1) 数据标签

数据标签即为数据标志提供附加信息的标签，数据标签代表源于数据表单元格的单个数据点或值。在设置数据标签前，首先需要设置该图表应用数据标签，否则程序报错。其语法形式为：

```
Chart.OleProcedure("ApplyDataLabels"); //设置应用数据
//标签
```

说明：当设置应用数据标签后，各数据标志默认为显示“值”，当对不同的数据标志应用不同的数据标签时，数据标志更改为相应的数据标签选项。

数据标签各选项及对应的变量名称如表 18 所示。

表 18 数据标签类型及取值

变量	类型
ShowSeriesName	系列名称
ShowCategoryName	类别名称
ShowValue	值
ShowPercentage	百分比
ShowBubbleSize	气泡尺寸
Separator	分隔符
ShowLegendKey	图例项标示

设置数据标签，分为 3 个步骤：

第一步，选定待设置的数据系列，通过“SeriesCollection”属性获取指定的数据系列，1 表示第一个系列的数据标志，2 表示第二个系列的数据标志，依此类推。

第二步，通过“DataLabels”属性获取该数据系列的数据标签属性。

第三步，设置具体的数据标签选项，true 表示选中该选项，false 则不选。

下面的代码实现了对第一个系列的数据标志设置“类别名称”标签的功能。

```
Chart.OlePropertyGet ("SeriesCollection",1).
OlePropertyGet("DataLabels").OlePropertySet("ShowCategory
Name",true);
```

(2) 分隔符

当数据标签设置了多个选项之后，通常在各选项之间添加分隔符加以区分。其语法形式为：

```
Chart.OlePropertyGet("SeriesCollection",1).OleProperty
Get("DataLabels").OlePropertySet("Separator",",");
```

上述代码表示用“,”区分各数据标签选项。

(3) 图例项标示

设置图例项标示的前提是指定的数据标志已选择了“值”数据标签，否则程序报错。其语法形式为：

```
Chart.OlePropertyGet ("SeriesCollection",1).
OlePropertyGet("DataLabels").OlePropertySet("ShowLegendKe
y",true);
```

2.3.6 数据表

图表选项的数据表设置如图 8 所示。

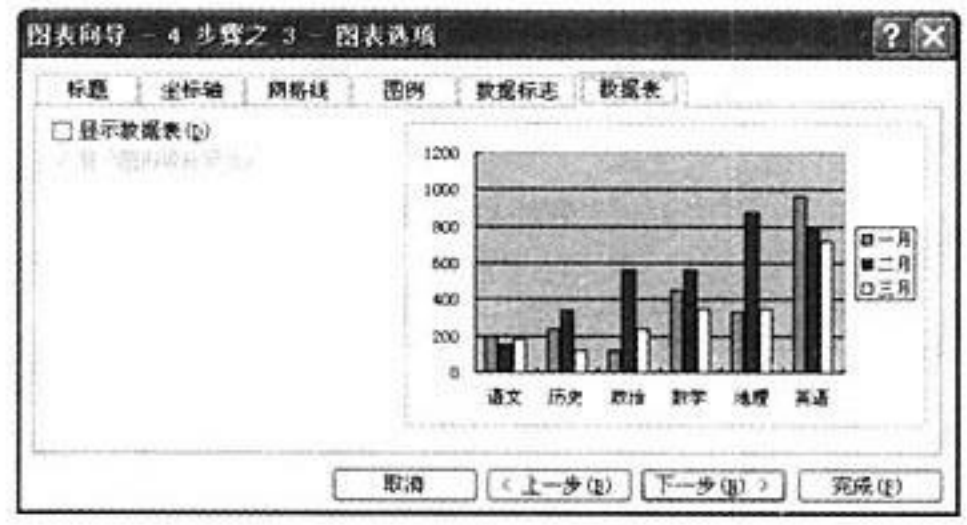


图 8 图表选项之数据表

(1) 显示数据表

通过“HasDataTable”属性设置是否在图表中显示数据表，该属性为 true 表示显示数据表，false 则不显示。

```
Chart.OlePropertySet("HasDataTabel",true);
```

(2) 显示图例项标示

选择显示数据表后，通过“ShowLegendKey”属性，便可设置是否在数据表中显示图例项标示，true 表示显示图例项标示，false 则不显示。

```
Chart.OlePropertyGet ("DataTable").OlePropertySet ("
ShowLegendKey",false);
```

2.4 图表位置

图表位置设置如图 9 所示。

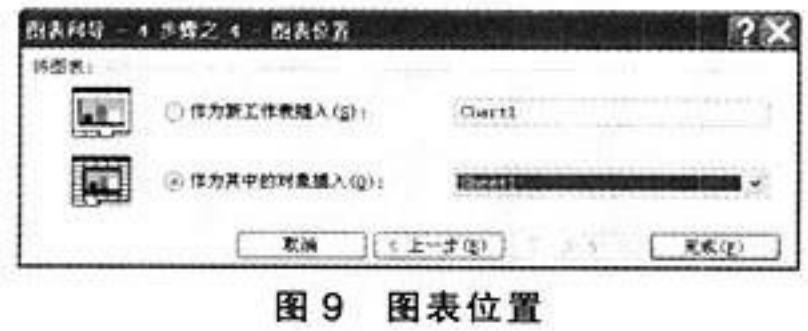


图 9 图表位置

图表在工作表中有两种存在方式：

一种是嵌入式图表。当希望图表作为工作表的一部分，与数据或其他图表在一起时，嵌入式图表是最好的选择。

第二种是图表工作表。当希望图表显示最大尺寸，而且不会妨碍数据或其他图表时，使用图表工作表。图表工作表的位置是固定的，其尺寸取决于工作表的尺寸。

通过“Location”属性来设置图表的位置，其语法形式分别为：

```
//图表工作表
Chart.OlePropertySet("Location",xlLocationAsNewSheet);
//嵌入式图表
Chart.OlePropertySet ("Location",xlLocationAsObject,
Sheet.OlePropertyGet("Name"));
```

说明：当添加嵌入式图表时，需要指定工作表的名称。

3 实现方法

下面结合具体实例，说明定制图表的操作方法。现有文件“D:\book1.xls”，其内容如图 10 所示。

	A	B	C	D
1	教材销售表			
2	书名	一月	二月	三月
3	语文	200	150	180
4	历史	234	340	120
5	政治	120	560	235
6	数学	450	560	345
7	地理	330	876	346
8	英语	960	800	720

图 10 教材销售表

生成图表的代码如下：

```
void GenerateChart()
{
 Variant ExcelApp,Workbook,Sheet,Chart,Range;
 try
    {
     ExcelApp=Variant::CreateObject("Excel.Application");
//创建 Excel 应用程序对象
    }
    catch(...)
    {
      //如果系统未安装 Excel,则显示警告信息
      MessageDlg("无法启动 Excel！",mtError,TMsgDlgButt
ons() << mbOK, 0);
      return;
   }
//设置应用程序不显示
ExcelApp.OlePropertySet("Visible",true);
AnsiString strFileName;
strFileName="d:\\book1.xls";
//打开指定的工作簿
ExcelApp.OlePropertyGet("Workbooks").OleFunction("Open",
strFileName.c_str());
 //获取当前工作簿对象
 Workbook=ExcelApp.OlePropertyGet("ActiveWorkbook");
 //获取当前工作表对象
 Sheet=Workbook.OlePropertyGet("ActiveSheet");
 //添加图表对象
  Chart =ExcelApp.OlePropertyGet  ("Charts").OleFunction ("
Add");
 //设置图表类型,这里设置的为簇状柱形图
 Chart.OlePropertySet("ChartType",51);
 //指定图表的数据区域
 Range=Sheet.OlePropertyGet("Range","A2:D8");
 //设置图表按行进行排列
 Chart.OleProcedure("SetSourceData",Range,xlRows);
 //图表标题
 Chart.OlePropertySet("HasTitle",true);
 AnsiString strTitle="教材销售情况图";
  Chart.OlePropertyGet  ("ChartTitle").OlePropertySet ("Text",
strTitle.c_str());
 //设置分类轴标题
      Chart.OlePropertyGet      ("Axes",xlCategory,xlPrimary).
OlePropertySet("HasTitle",true);
      Chart.OlePropertyGet      ("Axes",xlCategory,xlPrimary).
OlePropertyGet("AxisTitle").OlePropertySet("Text","书籍名称");
 //设置数值轴标题
   Chart.OlePropertyGet   ("Axes",xlValue).OlePropertySet  ("
HasTitle",true);
   Chart.OlePropertyGet   ("Axes",xlValue).OlePropertyGet  ("
AxisTitle").OlePropertySet("Text","销售量");
 //设置坐标轴以时间刻度方式显示
  Chart.OlePropertyGet  ("Axes",xlCategory).OlePropertySet ("
CategoryType",3);
 //设置显示分类轴和数值轴的主要网格线和次要网格线
  Chart.OlePropertyGet  ("Axes",xlCategory).OlePropertySet ("
HasMajorGridLines",true);
  Chart.OlePropertyGet  ("Axes",xlCategory).OlePropertySet ("
HasMinorGridLines",true);
   Chart.OlePropertyGet   ("Axes",xlValue).OlePropertySet  ("
HasMajorGridLines",true);
   Chart.OlePropertyGet   ("Axes",xlValue).OlePropertySet  ("
HasMinorGridLines",true);
 //设置图例位置为靠上
 //Chart.OlePropertySet("HasLegend",false); //不显示图例,默
//认情况下显示图例
  Chart.OlePropertyGet("Legend").OlePropertySet("Position",-
4160);
 //设置应用数据标签
 Chart.OleProcedure("ApplyDataLabels");
 //设置第一个数据标志的数据标签显示分类名称和值
  Chart.OlePropertyGet("SeriesCollection",1).OlePropertyGet("
DataLabels").OlePropertySet("ShowCategoryName",true);
```

(下转第 40 页)

C# 实现即时通信软件

李福红 姜秀宇

摘 要：通过对当前流行的即时通信软件的工作原理进行分析，并利用 C# 语言编程实现一个类似 QQ 的聊天程序，进而讲解网络编程中的协议使用。

关键词：C# 语言；即时通信软件；网络编程

很多人接触网络，是从腾讯 QQ 开始的。一个好的工具软件可以有这么大的影响，让人倍感欣慰。做软件开发需要好的软件，如：腾讯 QQ 的开发思路。为了让大家更好地理解平时常见的软件——QQ 的工作原理，在下文中将利用网络软件开发的相关知识来实现一个类似 QQ 的聊天程序。

1 即时通信系统

在生活中经常使用即时通信的软件，经常接触到的有：QQ、阿里旺旺、MSN 等。这些都是属于即时通信（Instant Messenger，IM）软件，IM 是指所有能够即时发送和接收互联网消息的软件。

在前面专题 P2P 编程中介绍过 P2P 系统分两种类型——单纯型 P2P 和混合型 P2P（QQ 就是属于混合型的应用），混合型 P2P 系统中的服务器（也叫索引服务器）起到协调的作用。在文件共享类应用中，如果采用混合型 P2P 技术的话，索引服务器就保存着文件信息，这样就可能会造成版权的问题，然而在即时通信类的软件中，因为客户端传递的都是简单的聊天文本而不是网络媒体资源，这样就不存在版权问题了，在这种情况下，就可以采用混合型 P2P 技术来实现即时通信软件。前面已经讲了，腾讯的 QQ 就是属于混合型 P2P 的软件。

因此本专题要实现一个类似 QQ 的聊天程序，其中用到的 P2P 技术是属于混合型 P2P，而不是前一专题中的采用的单纯型 P2P 技术，同时本程序的实现也会用到 TCP、UDP 编程技术。具体的相关内容大家可以搜索相关的资料。

2 系统设计

本程序采用 P2P 方式，各个客户端之间直接发消息进行聊天，服务器在其中只是起到协调的作用，下面先理清程序的流程：

2.1 程序流程设计

当一个新用户通过客户端登录系统后，从服务器获取在线的用户信息列表，列表信息包括系统中每个用户的地址，然后用户就可以单独向其发消息。如果有用户加入或者在线用户退出时，服务器就会及时发消息通知系统中的所有其他客户端，达到它们即时地更新用户信息列表。

根据上面大致的描述，可以把系统的流程分为下面几步来理解（大家可以参考 QQ 程序将会更好地理解本程序的流程）：

（1） 用户通过客户端进入系统，向服务器发出消息，请求登录。

（2） 服务器收到请求后，向客户端返回回应消息，表示同意接受该用户加入，并把自己（指的是服务器）所在监听的端口发送给客户端。

（3） 客户端根据服务器发送过来的端口号和服务器建立连接。

（4） 服务器通过该连接把在线用户的列表信息发送给新加入的客户端。

（5） 客户端获得了在线用户列表后就可以自己选择在线用户聊天。（程序中另外设计一个类似 QQ 的聊天窗口来进行聊天）

（6） 当用户退出系统时也要及时通知服务器，服务器再把这个消息转发给每个在线的用户，使客户端及时更新本地的用户信息列表。

2.2 通信协议

所谓协议就是约定，即服务器和客户端之间会话信息的内容格式进行约定，使双方都可以识别，达到更好的通信。

2.2.1 客户端和服务器之间的对话

（1）登录过程

① 客户端用匿名 UDP 的方式向服务器发出下面的信息：

login, username, localIPEndPoint

消息内容包括 3 个字段，每个字段用 “,” 分割，login 表示的是请求登录；username 表示用户名；localIPEndPint 表示客户端本地地址。

② 服务器收到后以匿名 UDP 返回下面的回应：

Accept, port

其中 Accept 表示服务器接受请求，port 表示服务器所在的端口号，服务器监听着这个端口的客户端连接。

③ 连接服务器，获取用户列表

客户端从上一步获得了端口号，然后向该端口发起 TCP 连接，向服务器索取在线用户列表，服务器接受连接后将用户列表传输到客户端。用户列表信息格式如下：

username1,IPEndPoint1;username2,IPEndPoint2;...;end

username1、username2 表示用户名，IPEndPoint1, IPEndPoint2 表示对应的端点，每个用户信息都是由" 用户名+端点" 组成，用户信息以 ";" 隔开，整个用户列表以 "end" 结尾。

(2) 注销过程

用户退出时，向服务器发送如下消息：

logout,username,localIPEndPoint

这条消息看字面意思大家都知道就是告诉服务器 username+localIPEndPoint 这个用户要退出了。

2.2.2 服务器管理用户

(1) 新用户加入通知

因为系统中在线的每个用户都有一份当前在线用户表，因此当有新用户登录时，服务器不需要重复地给系统中的每个用户再发送所有用户信息，只需要将新加入用户的信息通知其他用户，其他用户再更新自己的用户列表。

服务器向系统中每个用户广播如下信息：

login,username,remoteIPEndPoint

在这个过程中服务器只是负责将收到的" login" 信息转发出去。

(2) 用户退出

与新用户加入一样，服务器将用户退出的消息进行广播转发：

logout,username,remoteIPEndPoint

2.2.3 客户端之间聊天

用户进行聊天时，各自的客户端之间是以 P2P 方式进行工作的，不与服务器有直接联系，这也是 P2P 技术的特点。

聊天发送的消息格式如下：

talk, longtime, selfUserName, message

其中，talk 表明这是聊天内容的消息；longtime 是长时间格式的当前系统时间；selfUserName 为发送发的用户名；message 表示消息的内容。

协议设计介绍完后，下面就进入本程序的具体实现的介绍。

注：协议是本程序的核心，也是所有软件的核心，每个软件产品的协议都是不一样的，QQ 有自己的一套协议，MSN 又有另一套协议，所以使用的 QQ 的用户无法和用 MSN 的朋友进行聊天。

3 程序实现

服务器端核心代码：

```
// 启动服务器
// 根据博客中协议的设计部分
// 客户端先向服务器发送登录请求,然后通过服务
//器返回的端口号
// 再与服务器建立连接
// 所以启动服务按钮事件中有两个套接字:一个是
//接收客户端信息套接字和
// 监听客户端连接套接字
  private void btnStart_Click (object sender,
EventArgs e)
{
// 创建接收套接字
serverIp = IPAddress.Parse(txbServerIP.Text);
 serverIPEndPoint = new IPEndPoint(serverIp,
int.Parse(txbServerport.Text));
      receiveUdpClient = new UdpClient(serverI
PEndPoint);
// 启动接收线程
  Thread receiveThread = new Thread(Receive
Message);
receiveThread.Start();
btnStart.Enabled = false;
btnStop.Enabled = true;
// 随机指定监听端口
Random random = new Random();
tcpPort = random.Next(port + 1, 65536);
// 创建监听套接字
tcpListener = new TcpListener(serverIp, tcpPort);
tcpListener.Start();
// 启动监听线程
Thread listenThread = new Thread(ListenClient
Connect);
listenThread.Start();
AddItemToListBox(string.Format("服务器线程{0}
启动,监听端口{1}",serverIPEndPoint,tcpPort));
}
// 接收客户端发来的信息
private void ReceiveMessage()
{
 IPEndPoint remoteIPEndPoint = new IPEndPoint
(IPAddress.Any, 0);
while (true)
{
  try
  {
// 关闭 receiveUdpClient 时下面一行代码
//会产生异常
byte [] receiveBytes = receiveUdpClient.
Receive(ref remoteIPEndPoint);
 string message = Encoding.Unicode.
GetString(receiveBytes, 0, receiveBytes.Length);
// 显示消息内容
AddItemToListBox(string.Format("{0}:{1}",
```

```
remoteIPEndPoint,message));
                    // 处理消息数据
                     // 根据协议的设计部分,从客户端发送来
//的消息是具有一定格式的
                    // 服务器接收消息后要对消息做处理
                    string[] splitstring = message.Split(',');
                    // 解析用户端地址
            string[] splitsubstring = splitstring[2].Split(':');
                     IPEndPoint clientIPEndPoint = new
IPEndPoint (IPAddress.Parse (splitsubstring [0]), int.Parse
(splitsubstring[1]));
                    switch (splitstring[0])
                    {
                      // 如果是登录信息,向客户端发送应答
//消息和广播有新用户登录消息
                      case "login":
      User user = new User(splitstring[1], clientIPEndPoint);
                        // 往在线的用户列表添加新成员
                        userList.Add(user);
                         AddItemToListBox (string.Format("
用户{0}({1})加入", user.GetName(), user.GetIPEndPoint()));
          string sendString = "Accept," + tcpPort.ToString();
                        // 向客户端发送应答消息
                        SendtoClient(user, sendString);
          AddItemToListBox (string.Format (" 向 {0} ({1}) 发 出 :
[{2}]", user.GetName(), user.GetIPEndPoint(), sendString));
                for (int i = 0; i < userList.Count; i++)
                        {
                 if (userList[i].GetName() ! = user.GetName())
                        {
                 // 给在线的其他用户发送广播消息
                            // 通知有新用户加入
                SendtoClient(userList[i], message);
                        }
                      }
    AddItemToListBox(string.Format("广播:[{0}]", message));
                        break;
                      case "logout":
                for (int i = 0; i < userList.Count; i++)
                        {
          if (userList[i].GetName() == splitstring[1])
                        {
                              AddItemToListBox (string.
Format (" 用 户 {0} ({1}) 退 出 ",userList [i].GetName (),userList [i].
GetIPEndPoint()));
                userList.RemoveAt(i); // 移除用户
                            }
                        }
            for (int i = 0; i < userList.Count; i++)
                        {
                            // 广播注销消息
            SendtoClient(userList[i], message);
                                                    广播:[{0}]", message));
                                                    }
                                                    AddItemToListBox (string.Format("
                                                    break;
                                                    }
                                                }
                                            }
                                            catch
                                            {
                                                // 发送异常退出循环
                                                break;
                                            }
                                        }
                                        AddItemToListBox(string.Format("服务线程{0}终
止", serverIPEndPoint));
                                    }
                                    // 向客户端发送消息
                                private void SendtoClient(User user, string message)
                                    {
                                        // 匿名方式发送
                                        sendUdpClient = new UdpClient(0);
                                         byte [] sendBytes = Encoding.Unicode.
GetBytes(message);
                                    IPEndPoint remoteIPEndPoint =user.GetIPEndPoint();
                                         sendUdpClient.Send (sendBytes,sendBytes.
Length,remoteIPEndPoint);
                                        sendUdpClient.Close();
                                    }
                                    // 接受客户端的连接
                                    private void ListenClientConnect()
                                    {
                                        TcpClient newClient = null;
                                        while (true)
                                        {
                                            try
                                            {
                                      newClient = tcpListener.AcceptTcpClient();
                                                AddItemToListBox(string.Format("接受客
户端{0}的 TCP 请求",newClient.Client.RemoteEndPoint));
                                            }
                                            catch
                                            {
                                                AddItemToListBox(string.Format("监听线
程({0}:{1})", serverIp, tcpPort));
                                                break;
                                            }
                                        Thread sendThread = new Thread(SendData);
                                                sendThread.Start(newClient);
                                        }
                                    }
                                    // 向客户端发送在线用户列表信息
                                    // 服务器通过 TCP 连接把在线用户列表信息发送给
//客户端
```

```
        private void SendData(object userClient)
        {
    TcpClient newUserClient = (TcpClient)userClient;
            userListstring = null;
            for (int i = 0; i < userList.Count; i++)
            {
                userListstring += userList[i].GetName() + ","
      + userList[i].GetIPEndPoint().ToString() + ";";
            }
            userListstring += "end";
            networkStream = newUserClient.GetStream();
          binaryWriter = new BinaryWriter(networkStream);
            binaryWriter.Write(userListstring);
            binaryWriter.Flush();
             AddItemToListBox (string.Format (" 向 {0} 发 送
[{1}]", newUserClient.Client.RemoteEndPoint, userListstring));
            binaryWriter.Close();
            newUserClient.Close();
        }
```

客户端核心代码：

View Code

```
    // 登录服务器
  private void btnlogin_Click(object sender, EventArgs e)
        {
            // 创建接受套接字
      IPAddress clientIP = IPAddress.Parse(txtLocalIP.Text);
                clientIPEndPoint = new IPEndPoint (clientIP,
int.Parse(txtlocalport.Text));
      receiveUdpClient = new UdpClient(clientIPEndPoint);
                // 启动接收线程
      Thread receiveThread = new Thread(ReceiveMessage);
                receiveThread.Start();
                // 匿名发送
                sendUdpClient = new UdpClient(0);
                // 启动发送线程
    Thread sendThread = new Thread(SendMessage);
                sendThread.Start (string.Format ("login,{0},{1}",
txtusername.Text, clientIPEndPoint));
                btnlogin.Enabled = false;
                btnLogout.Enabled = true;
                this.Text = txtusername.Text;
            }
            // 客户端接受服务器回应消息
            private void ReceiveMessage()
            {
                IPEndPoint remoteIPEndPoint = new
IPEndPoint(IPAddress.Any,0);
                while (true)
                {
                    try
                    {
                        // 关闭 receiveUdpClient 时会产生异常
                        byte [] receiveBytes = receiveUdpClient.
Receive(ref remoteIPEndPoint);
                        string message = Encoding.Unicode.
GetString(receiveBytes,0,receiveBytes.Length);
                        // 处理消息
                    string[] splitstring = message.Split(',');
                        switch (splitstring[0])
                        {
                            case "Accept":
                            try
                            {
                                tcpClient = new TcpClient();
                                    tcpClient.Connect
(remoteIPEndPoint.Address, int.Parse(splitstring[1]));
                                if (tcpClient ! = null)
                                {
                                    // 表示连接成功
                        networkStream = tcpClient.GetStream();
                    binaryReader = new BinaryReader(networkStream);
                                }
                            }
                            catch
                            {
                        MessageBox.Show("连接失败", "异常");
                            }
                    Thread getUserListThread = new Thread(GetUserList);
                                getUserListThread.Start();
                                break;
                            case "login":
                     string userItem = splitstring[1] + "," + splitstring[2];
                                AddItemToListView(userItem);
                                break;
                            case "logout":
                    RemoveItemFromListView(splitstring[1]);
                                break;
                            case "talk":
                     for (int i = 0; i < chatFormList.Count; i++)
                                {
                     if (chatFormList[i].Text == splitstring[2])
                                    {
                                        chatFormList [i].ShowTalkInfo
(splitstring[2], splitstring[1], splitstring[3]);
                                    }
                                }
                                break;
                        }
                    }
                    catch
                    {
                        break;
                    }
```

```
    }
}
// 从服务器获取在线用户列表
private void GetUserList()
{
    while (true)
    {
        userListstring = null;
        try
        {
userListstring = binaryReader.ReadString();
            if (userListstring.EndsWith("end"))
            {
string[] splitstring = userListstring.Split(';');
for (int i = 0; i < splitstring.Length - 1; i++)
                {
                    AddItemToListView(splitstring[i]);
                }
                binaryReader.Close();
                tcpClient.Close();
                break;
            }
        }
        catch
        {
            break;
        }
    }
// 发送登录请求
    private void SendMessage(object obj)
    {
        string message = (string)obj;
byte[] sendbytes = Encoding.Unicode.GetBytes(message);
IPAddress remoteIp = IPAddress.Parse(txtserverIP.Text);
                    IPEndPoint remoteIPEndPoint = new
IPEndPoint(remoteIp, int.Parse(txtServerport.Text));
            sendUdpClient.Send (sendbytes, sendbytes.
Length, remoteIPEndPoint);
            sendUdpClient.Close();
        }
```

程序运行结果：

首先运行服务器窗口，在服务器窗口点击“启动”按钮来启动服务器，然后客户端首先指定服务器的端口号，修改用户名（这里也可以不修改，使用默认的也可以），然后点击“登录”按钮来登录服务器（也就是告诉服务器本地的客户端地址），然后从服务器端获得在线用户列表，界面演示如图 1、图 2 所示。

然后用户可以双击在线用户进行聊天（此程序支持与多人进行聊天），功能演示如图 3 所示。

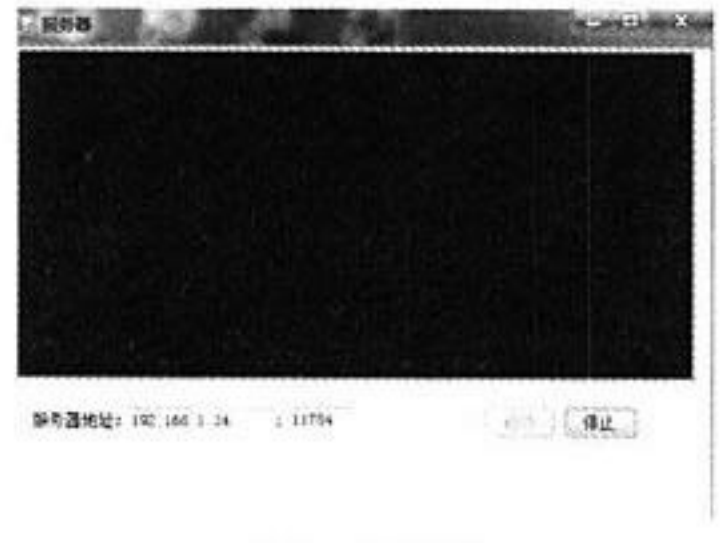


图 1　服务端

图 2　客户端的在线用户列表

图 3　客户端聊天窗口

双方进行聊天时，这里没有实现像 QQ 一样，有人发信息来在对应的客户端就有消息提醒的功能，所以双方进行聊天的过程中，每个客户端都需要在在线用户列表中点击聊天的对象来激活聊天对话框（意思就是从图片中可以看出“天涯”客户端想和剑痴聊天的话，就在“在线用户”列表双击剑痴来激活聊天窗口，同时“剑痴”客户端也必须双击“天涯”来激活聊天窗口，这样双方就看到对方发来的信息了，（不激活窗口，即使发送了信息，只是没有一个窗口来进行显示）），而且从图片中也可以看出——此程序支持与多人聊天，即天涯同时与“剑痴”和“大地”同时聊天。

4　结语

介绍了实现一个类似 QQ 的聊天程序，让大家可以熟悉网络开发相关技术，也让大家更好地理解即时通信软件（腾讯 QQ）的工作原理和软件协议的设计。

（收稿日期：2012-11-06）

Excel在工资查询系统中的运用

许 国

摘　要：利用VBA开发低成本、高效率的学校工资查询系统，具体有很高的借鉴意义和实用价值。

关键词：VBA编程；查询系统；办公自动化

随着财政改革的进一步深入，学校单位的工资都纳入了财政统发工资软件系统，单位只要把工资总额打入到指定的银行账户，再由银行打入个人工资卡中，单位职工一般只知道一个工资总数，对于具体的工资明细要到财务室查询才能详细了解。鉴于此，我们运用Excel开发了工资查询系统，方便了学校教职工的查询，也节约了一定的办公经费。下面详细介绍工资查询系统的开发过程。

1　下载工资发放表

从财政的统发工资软件系统中下载工资发放明细表，保存为Excel工作簿格式，并命名为“工资发放表”，工资发放表样式如图1所示，工资表表头只能设置为一行。

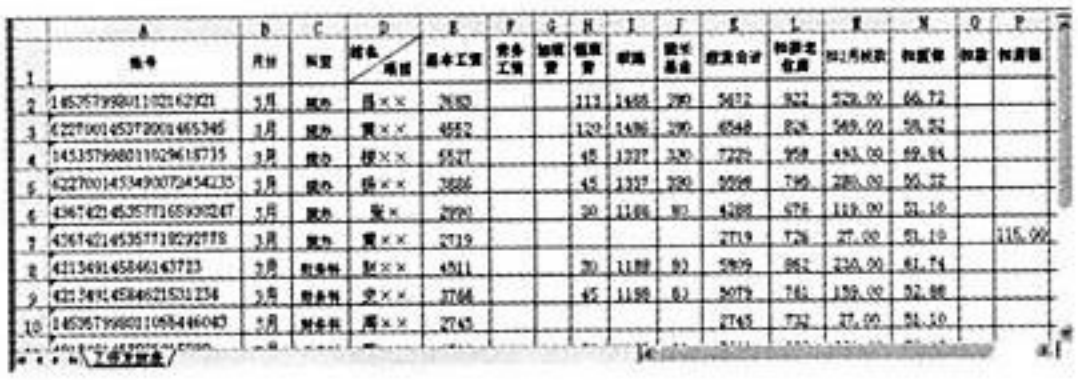


图1　工资发放表样式

2　创建工资查询系统客户端

创建一个“工资查询系统客户端”工作簿，在A5单元格中创建两个窗体按钮，分别命名为“查询”和“清空”，完成后的内容如图2所示，其中A4单元格用于输入要查询的代发工资个人银行账号，第6行内容与工资发放表中的表头一致。

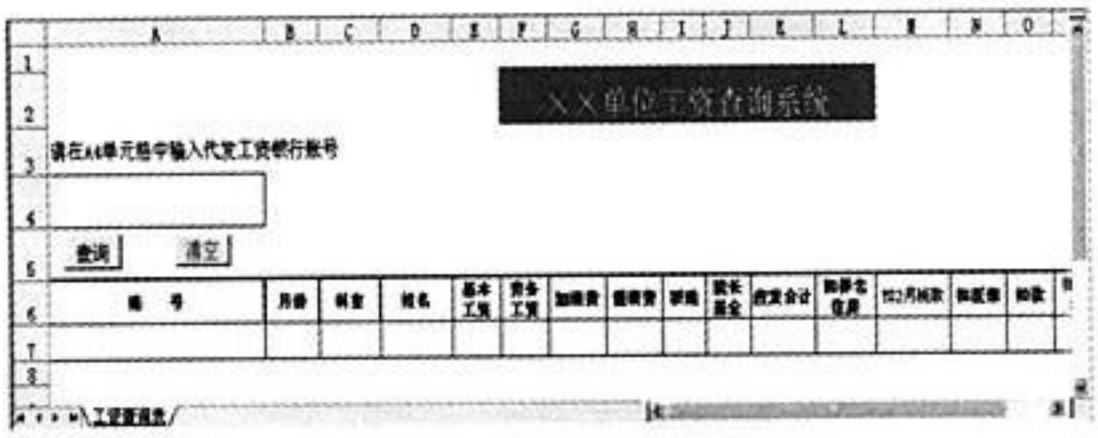


图2　创建窗体按钮

3　编写宏代码，实现自动化

（1）打开“工资查询系统客户端”工作簿，选择“工具”→“宏”→“Visual Basic编辑器”菜单，进入VBA编辑状态，执行“插入”→“模块”命令，插入一个模块1，在编辑窗口中输入如下代码：

```
Sub 工资明细查询()
  Application.ScreenUpdating = False
  On Error Resume Next
  Dim mystr As String
  mystr = Cells(4, 1)
   Shell "net use \\144.36.61.53 /user:administrator 123456",
vbHide ′连接服务器
  Set Conn = CreateObject("adodb.connection")    ′设置对象
   Conn.Open "provider =microsoft.jet.oledb.4.0;extended
properties=excel 8.0;data _ source=\\144.36.61.53\工资查询
系统\工资发放表.xls"
  Sql = "select *  from [工资发放表$] where [工资发放表$].账
号= '" + mystr + "' "
  Range("A7").CopyFromRecordset Conn.Execute(Sql)
   If Range("a7") = "" Then
     MsgBox "您输入的账号有误或未能连接到服务器,请核实"
   End If
  Range("a4").Select
  Conn.Close       ′关闭链接
  Set Conn = Nothing ′释放对象变量
  Application.ScreenUpdating = True
End Sub
```

代码说明：以上代码是先与服务器（代码中的144.36.61.53是服务器的IP地址，administrator为服务器的登录用户名，123456为登录密码）中的“工资发放表”建立连接，然后查找与A4单元格中账号相同的职工的工资明细，并复制到工资查询表的第7行中。

（2）插入一个模块2，在编辑窗口中输入如下代码：

```
Sub 清空()
  Set Sh1 = Worksheets("工资查询表")
  Sh1.Range(Cells(7, 1), Cells(7, 20)).ClearContents
   Sh1.Range("a4").ClearContents
```

End Sub

代码说明：以上代码用于清空查询表中的工资明细信息。

由于上述代码是存放在客户端的工作簿中的，而客户端是分发给每位教职工的，因此要对源代码进行保护，在代码编辑窗口中打开“工具”→“Project 属性”窗口，选取“保护”选项，选中“查看时锁定工程”选项，然后输入密码，按确定即可实现 VBA 代码保护，如图 3 所示。

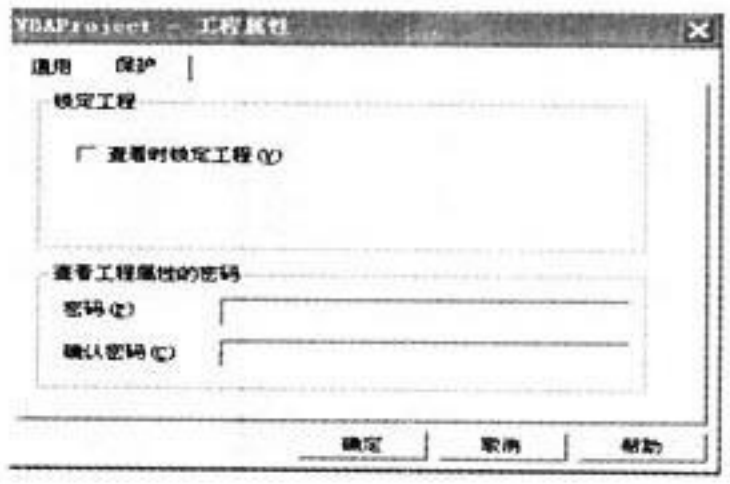


图 3 VBA 代码保护

4 进行明细查询

(1) 选中“查询”窗体按钮，单击右键，选择“指定宏”命令，弹出“宏”对话框，选中宏“工资明细查询”，单击对话框右上方的“确定”按钮，用同样的方法把“清空”宏指定“清空”窗体按钮。

(2) 以后要查询教职工工资明细时，只要打开“工资查询系统客户端”工作簿，在客户端工作簿的 A4 单元格中输入代发工资个人银行账号，按“查询”按钮即可查到教职工的工资明细，按“清空”按钮，清空查询结果。

5 相关要求

(1) “工资发放表”工作簿用于存放所有教职工的工资明细，此工作簿作为数据库存放到电脑中的“工资查询系统”文件夹中，此文件夹应设置为共享文件夹，“工资发放表”工作簿放在局域网的一台电脑中即可，且要保证其他电脑能够访问这台电脑。

(2) “工资查询系统客户端”工作簿要分发给每一位教职工，或者放到网上供教职工下载，教职工电脑中 Excel 工作簿中的宏的安全性要设置为“中”或“低”。

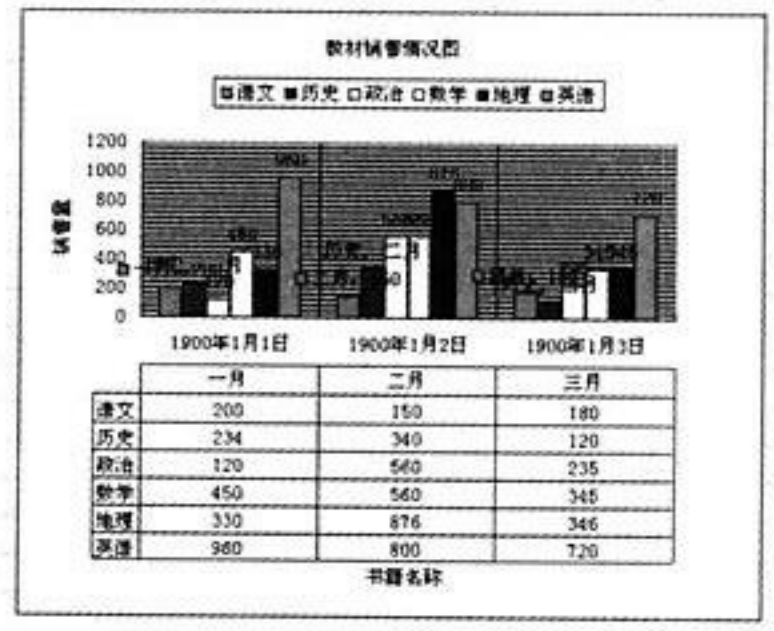
（收稿日期：2012-05-29）

（上接第 33 页）

```
  Chart.OlePropertyGet("SeriesCollection",1).OlePropertyGet("
DataLabels").OlePropertySet("ShowValue",true);
  //设置第一个数据标志的数据标签以","进行分隔
  Chart.OlePropertyGet("SeriesCollection",1).OlePropertyGet("
DataLabels").OlePropertySet("Separator",",");
  //设置第一个数据标志的数据标签显示图例项标示
  Chart.OlePropertyGet("SeriesCollection",1).OlePropertyGet("
DataLabels").OlePropertySet("ShowLegendKey",true);
  //设置第二个数据标志的数据标签显示系列名称
  Chart.OlePropertyGet("SeriesCollection",2).OlePropertyGet("
DataLabels").OlePropertySet("ShowSeriesName",true);
  //显示数据表
  Chart.OlePropertySet("HasDataTable",true);
  //数据表中不显示图例项标示
    Chart.OlePropertyGet ("DataTable").OlePropertySet ("
ShowLegendKey",false);
  //添加嵌入式图表
    Chart.OleProcedure ("Location",xlLocationAsObject,Sheet.
OlePropertyGet("Name"));
  //设置图表的位置和长宽
    //设定工作表中的第一个图表,如果是第二个图表,用 Sheet.
//OlePropertyGet("ChartObjects",2)
  Sheet.OlePropertyGet ("ChartObjects",1).OlePropertySet ("
Top",150);
  Sheet.OlePropertyGet ("ChartObjects",1).OlePropertySet ("
Left",20);
  Sheet.OlePropertyGet ("ChartObjects",1).OlePropertySet ("
Width",300);
  Sheet.OlePropertyGet ("ChartObjects",1).OlePropertySet ("
Height",250);
}
```

4 运行结果

在 C++ Builder 的事件处理程序中添加上述代码，程序执行后，将生成如图 11 所示形式的图表。



	一月	二月	三月
语文	200	150	180
历史	234	340	120
政治	120	560	235
数学	450	560	345
地理	330	876	346
英语	960	800	720

图 11 生成的图表

（收稿日期：2012-08-24）

用 Spring+JQuery 实现信息管理小系统

王玉贵

摘　要：分析了 Spring+JQuery 技术的实现细节，并通过一个系统实例，验证了该技术的高效和可行性。

关键词：Java 语言；网络编程；Spring+JQuery 技术

1　引言

Java 的世界真是一个万花筒，变化万端，五彩纷呈。在基于 Java 平台开发信息管理系统过程中，B/S 架构、MVC 模式已经变得非常热门，出现了一大批帮助人们实现这一目标的框架和技术，像 Hibernate、Toplink、Spring、Struts，真是八仙过海，各显其能。然而在框架选择上有些让人眼花缭乱、无所适从。于是先锋们披荆斩棘、探索出一条条道路，出现了像 SSH (Struts Spring Hibernate) 这样的组合拳套路。

如果不讲究方法，单单使用 JSP 和 servlet 技术就能进行 B/S 架构的系统开发。如果是那样，就会整天和 Html 标签与 Java 代码的混合体打交道，从而错过了 Java 世界的精彩。接下来将呈现的是一套框架组合使用技术：Spring+JQuery 组合技术，引领大家使用这个技术搭建一个很小的一个信息管理系统。虽然这不过是 Java 世界中的冰山一角，却也能够领略到 Java 的精彩。

2　介绍 Hello Word

Spring MVC 是服务器端 Java 框架，而 JQuery 是客户端 js 框架，如何进行组合应用呢。按编程界惯例，以 Hello Word 作为旅程的开始。图 1 就是程序界面，在输入框中输入姓名，点击“问候”按钮，在问候信息栏里就会显示问候信息。程序虽然简单，却能初步了解将要建立的这个小系统是如何工作的。

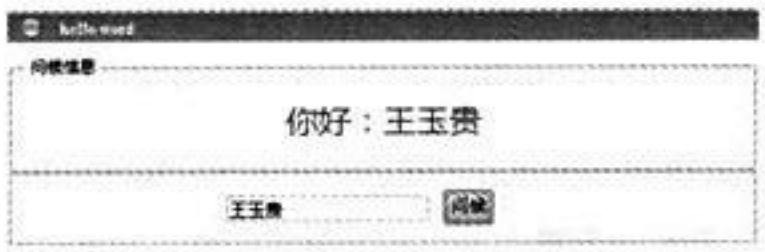


图 1　一个问候小程序

2.1　客户端代码

客户端使用静态的 html 页面加上 JavaScript 脚本，可以完全脱离 jsp 技术。这个组合与 Flex 技术中 MXML 和 ActionScript 组合相似，但是不依赖任何浏览器插件，具有更好的兼容性。下面就是 helloworld.htm 页面的结构，里面只列出了关键的标签。可以看出，它就是一个简单的静态 html 页面。

在这个页面中，首先引用了 jquery1.3.2 库和 jquery 表单插件代码，这是 js 工作的基础。接着引入 helloworld.js 脚本文件，注意命名规则是 html 页面及其相关的 js 脚本文件使用相同的名称。在 body 标签中定义了一个提交数据的表单和一个用来显示服务器端返回信息的 div 标签：

```
<html>
<head>  …
  <link  href = "resources/skin/blue/css/main.css"  rel = "
stylesheet" type="text/css">
  <script  type = "text/javascript"  src = "resources/js/jquery -
1.3.2.min.js"></script>
  <script  type = "text/javascript"  src = "resources/js/jquery.
form.js"></script>
  <script  type  =  "text/javascript"  src  =  "resources/js/
helloworld.js"></script>
</head>
<body> …<! –显示信息使用–->
  <div class="orgrootdiv" id="procmsgdiv" align="center"><
/div>
    …<! –提交信息使用表单–->
  <form action="data/system/sayHello" method="post" id="
helloform">
    <input  name = "yourname"  type = "text"  class = "
textfield">
    <input type="submit" name="button" value="问候">
</form>
…
</body>
</html>
```

在 helloworld.js 文件中，使用了一个 JQuery 函数$ (function () {})，在所有 js 文件中都使用了这样的结构，它的作用是当 html 页面在浏览器中下载完毕后，执行被传入的函数，相当于 body.ready () 函数。所有 js 代码都放在被传入函数的函数体中，它的结构如下：

```
$(function(){ //所有代码在这里 });
```

现在解释一下 helloworld.js 的代码。首先使用 jquery 最具特色、也是最常用的 dom 元素查找函数$ ('selector') 查找用于显示返回信息的 div 标签，把它缓存到一个变量中；接着查找页面中的表单元素，使用 jquery 表单插件把这个表单初始化成一个 Ajax 表单。这样做好处很多，最主要的是表单能够以 Ajax 的方式进行提交，不用刷新页面，并且能够方便地进行表单数据验证。

代码的功能就不多解释了，熟悉 jquery 的人都能看得出来，表单提交成功后，把服务器端返回的数据以调用参数的形式传给提交成功回调函数，回调函数得到执行，把数据显示在 div 标签里：

```
$(function(){
 var hlinfo=$('#procmsgdiv');
 $('#helloform').ajaxForm({beforeSubmit:beforsub,
                           dataType:'json',
                           success:function(r){
                                   hlinfo.
                                   html(r);
                                   }
                          });
 function beforsub(data){
     if(data.get('name','yourname').value==''){
    alert(' 请输入您的名字 ');
    return false;
     }
 }
});
```

2.2 服务器端代码

服务端充分运用了 Spring MVC 框架的功能，编写一个普通的 Java 类，也就是常说的 pojo (Plain Old Java Objects) 类，利用 Spring MVC 的 Annotation 注解功能就能使这个普通的 pojo 成为一个神奇的 MVC 控制器，完成交付给它的使命。

代码中使用了两个非常重要的 Annotation 注解，一个是 @Controller，就是它把一个 pojo 变成了一个 Spring MVC 控制器；另一个是 @RequestMapping，是这个注解把一个普通的类方法变成了一个可以通过 url 进行调用的控制器方法：

```
package cn.chinaunicom.sys.controller;
...
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.Request
Mapping;
import org.springframework.web.bind.annotation.Request
Method;
import org.springframework.web.servlet.ModelAndView;
@Controller
public class SystemController {
 @RequestMapping
    public ModelAndView sayHello(String yourname){
        return new ModelAndView ("jsonview","data"," 你
        好:"+yourname);
    }
}
```

2.3 联系的实现

客户端代码如何把数据提交给服务器端 sayHello 方法的呢？如果想彻底解释清楚这个过程，恐怕需要写一本书。在这里只想粗略地解释一下这个过程，使大家能够“知其然”，要想“知其所以然”，大家还需要找 Spring MVC 相关的书籍进一步学习。

一个来自客户端的 HTTP 请求在 Spring MVC 框架中经过多次流转，最后产生请求结果送回到客户端，这个过程如图 2 所示。

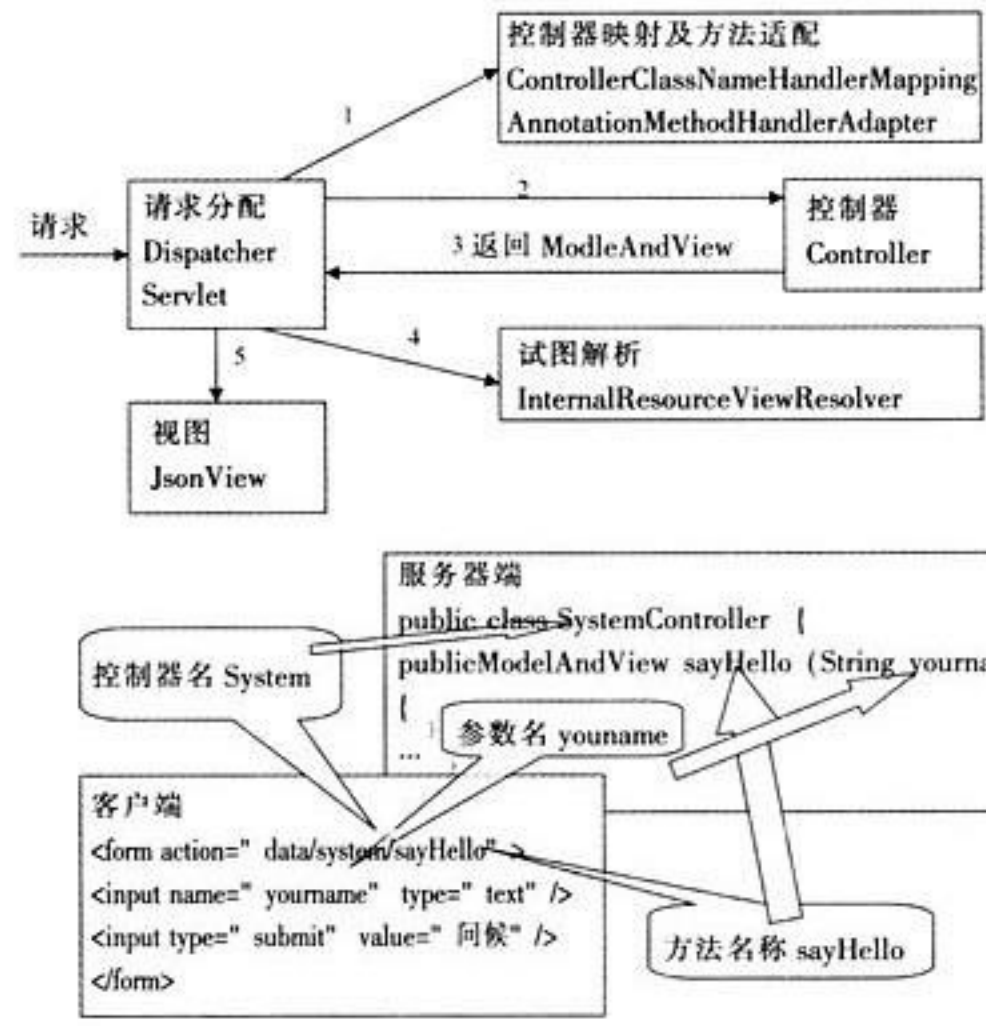


图 2 url 和 action 方法之间的对应关系

客户端调用服务器端控制器方法，主要用统一资源定位符 url 来确定对应关系。表单的 action 属性中填入了统一资源定位符，这个 url 分为 3 个部分，用“/”分隔。其中就包括控制器名称 (system，去掉 Controller 字样，不区分大小写) 和控制器方法 (syaHello，区分大小写) 两个部分，有了这两个部分就能定位到某一个控制器的某一个方法。有的 MVC 框架为了确定这个调用关系需要进行一大堆的配置，Spring MVC 采用了“约定优先于配置”的策略，使得 url 和 action 方法之间的对应关系使用命名约定来确定，几乎不用进行配置。

表单数据也使用了“约定优先于配置”的策略，只要客户端输入文本框的 name 属性和服务器端控制器方法的形参名称

一致，输入框中的数据就能传送到这个参数中来。

简单了解了源代码和 Spring MVC 的 url 和 action 方法之间的对应策略，理解 helloworld 就不难了。用户在文本输入框中输入姓名，点击“问候”按钮提交，根据表单 action 属性“data/system/sayHello”能够定位到 SystemController 的 sayHello 方法。该方法接收表单中名字为 username 的文本输入框的内容，在前面加上“你好”字样，转换成 json 格式的数据返回到客户端。客户端 javascript 回调提交成功函数，把结果显示在 div 标签中。

3 分工合作

上面介绍的 helloworld 比较简单，在服务器端一个控制器就可以完成任务，没有进行进一步分工。实际系统中往往要复杂得多，进行进一步分工就显得十分必要（如图 3 所示）。

图 3 各位员工的源代码位置

在系统中，已经进行了功能划分，就像一个公司中的员工一样，各部分各司其职，分工合作，共同完成公司的运营工作（如图 4 所示）。下面就带各位领导视察一下我们的员工。

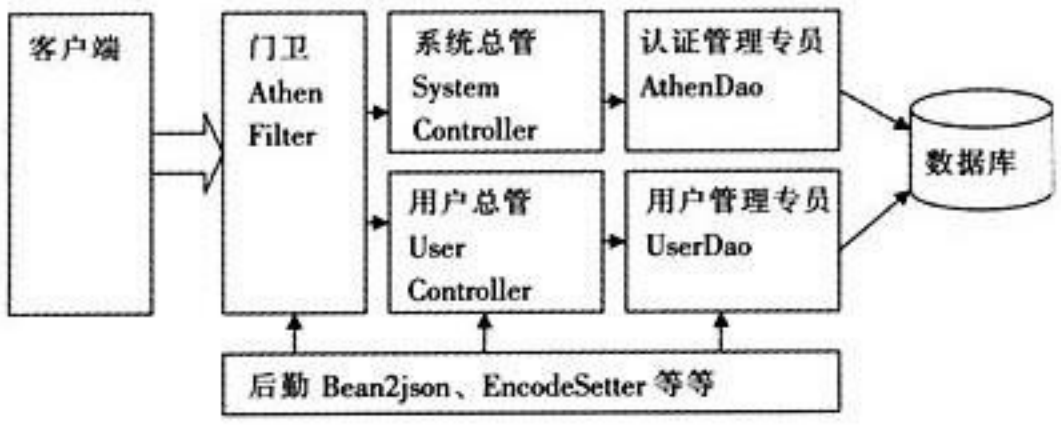


图 4 员工分工示意图

3.1 门卫 AuthenFilter

他的工作职责是把好大门，负责整个系统的安全工作。其实他就是一个过滤器，把没有通过用户认证的请求阻挡在大门以外。

3.2 接待员兼总管—各类控制器

如果客户端的请求能够通过门卫，下一个就传送到这里。它既像一个接待员，负责接收来自客户端的请求，又像是一位总管，负责分配工作，把工作交付到工人手里。工人完成工作后，向总管交付工作结果，由这位总管把工作结果返回给客户。

3.3 各类专员

各位专员分工明确，有的负责用户认证工作（AuthenDao），有的负责专业生产（UserDao）。的它具有专业技术水平很高，能够操作数据库，所有的业务逻辑都由它们掌握，能够很好地与总管配合，完成总管交付的工作，并及时地反馈工作结果。

3.4 后勤

这些员工是为其他员工服务的。它们虽然不抛头露面，但是其工作却非常重要。例如 EncodeSetter 负责文书抄写，使整个系统文字编码一致，不会出现烦人的乱码；Bean2Json 负责把 Java 对象转换成 json 格式的数据，以便客户端 Javascript 脚本进行处理。

4 实战--用户数据管理

有了前面的基础，接着可以进行一次实战演练，任务是进行用户数据的增删查改（CRUD）操作。目的是通过这次演练弄清各类员工如何进行配合的和工作流程，对系统进行进一步了解，最后达到能把这种模式熟练地应用到实际开发工作中去。参与用户数据工作员工如图 5 所示。

图 5 参与用户管理的员工

4.1 客户端代码

4.1.1 操作页面 User.htm

图 6 User.htm 运行时效果图

如图 6 所示的操作页面，只用一个静态 html 页面，用户数据增、删、查、改、列表都在同一个页面里完成。页面中设有查询表单、添加表单、修改表单 3 个表单，都使用 Ajax 方式与服务器端通信。添加表单和修改表单平时处于隐藏状态，点击相应的按钮才会显示出来。查询表单查询到数据后，交给用户列表控件进行列表显示。单击选择列表中的一个用户数据记录，可以对这个用户数据进行删除或者修改。

4.1.2 脚本 User.js

客户端如果没有 User.js 脚本文件，单靠 User.htm 页面本

身什么工作也不能完成。使用文中介绍的 spring+jquery 组合技术进行开发，服务器端不进行任何 html 处理，这部分工作完全转移到客户端用 Javascript 编程实现。因此 js 在系统中占有非常重要的位置。

由于源代码比较长，这里就不贴出来了，详细请看附加的源代码，有较为详细的注解。

4.2 服务器端代码

4.2.1 用户总管 UserController.java

```
package cn.chinaunicom.sys.controller;

import javax.annotation.Resource;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.Request
Mapping;
import org.springframework.web.servlet.ModelAndView;
import cn.chinaunicom.sys.dao.UserDao;
import cn.chinaunicom.sys.modle.User;

@Controller
public class UserController {
    @Resource(name="userdao")
    private UserDao userdao;
    @RequestMapping
    public ModelAndView delUser(String loginname){
        return new ModelAndView ("jsonview","data",
        userdao.delUser(loginname));
}
@RequestMapping
public ModelAndView addUser(User user){
    return new ModelAndView ("jsonview","data",
    userdao.addUser(user));
}
@RequestMapping
public ModelAndView modUser(User user){
    return new ModelAndView ("jsonview","data",
    userdao.modUser(user));
}
@RequestMapping
public ModelAndView getUsers(User user){
    return new ModelAndView ("jsonview","data",
        userdao.getUsers(user));
    }
}
```

4.2.2 用户管理专员 UserDao.java

```
package cn.chinaunicom.sys.dao;
import java.util.List;
import java.util.Map;
import javax.annotation.Resource;
import org.springframework.stereotype.Service;
import org.springframework.jdbc.core.simple.SimpleJdbc
Template;
import cn.chinaunicom.sys.modle.User;

@Service("userdao")
public class UserDao {
    private static final String addUser = "insert into user
    (loginname,password,realname) values(?,?,?)";
    private static final String modUser = "update user set
    realname=?,password=? where loginname=?";
    private static final String delUser = "delete from user
    where loginname=?";
    private static final String getUsers="select * from user
    where loginname like ? or loginname like ?";
    private static final String getAllUsers = "select * from
    user";
    @Resource(name = "db")
    private SimpleJdbcTemplate db;
    public List<Map<String,Object>> addUser(User user){
        db.update (addUser, user.getLoginname (),user.
        getPassword(),user.getRealname());
        return getAllUser();
    }
public List<Map<String,Object>> modUser(User user){
        db.update (modUser, user.getRealname (),user.
        getPassword(),user.getLoginname());
        return getAllUser();
    }
    public List <Map <String,Object >> delUser (String
loginname){
        db.update(delUser, loginname);
        return getAllUser();
    }
  public List<Map<String,Object>> getUsers(User user){
        return db.queryForList (getUsers, "% " +user.
        getRealname () + "% ","% " +user.getLoginname ()
        +"%");
    }
  private List<Map<String,Object>> getAllUser(){
    return db.queryForList(getAllUsers);
  }
}
```

5 Spring–JQuery 组合技术特点

5.1 干净的服务器端代码

在服务器端代码全部是层次清晰的 Java 类，专注于业务逻辑处理，完全不必考虑 html 用户界面视图处理，视图处理交由客户端 js 脚本去处理。这样做好处是服务器端编码人员可以脱离用户界面视图处理工作，有利于工程分工。

5.2 静态的 HTML 视图

用户视图使用静态的 HTML 页面，服务器端不用进行任何

处理，用户界面设计人员可以使用像 Dreamwever 这样的工具进行设计工作，不必考虑程序代码。

5.3 紧凑的 JSON 模型数据

服务器与客户端使用 JSON（JavaScript Object Notation）数据格式进行数据交换。与 XML 等其他数据交换格式相比，json 数据格式更加紧凑有效。同时，因为客户端采用 html+js 进行编程，并且用 json 数据格式与服务器端进行数据交换，这样可以做到客户用户界面与服务器端使用编程技术无关。也就是说，客户端编程人员不用关心服务器端技术，服务器端无论使用 Java、PHP、.NET 等不管哪一种技术进行编程，只要能够把数据转换成 json 数据格式，一切都解决了。

5.4 纯粹的 Ajax 请求

使用 Ajax 方式进行通信交互的好处想必大家也比较了解，第一个好处就是页面不用刷新，减轻服务器端负担。第二个好处是可以通过精心的用户界面设计来改善用户使用体验。一个好的用户界面设计可以通过页面分层、适时的隐藏和显示等技术，在一个页面中完成大量相关的功能，让用户使用体验提升到传统 C/S 架构程序的水平。

5.5 更彻底的 MVC 模式

Spring 框架提供的 MVC 模块可以很好地满足 MVC 软件架构设计，可是仍然需要引入 jsp 或者其他如 Velocity、freemarker 等表现（view）层技术，这些技术都需要依靠服务器端进行渲染处理，然后将模型数据（Modle）和用户界面（View）的混合物（html）推向客户端浏览器。服务器和网络的负担都很大 (如图 7 所示)。

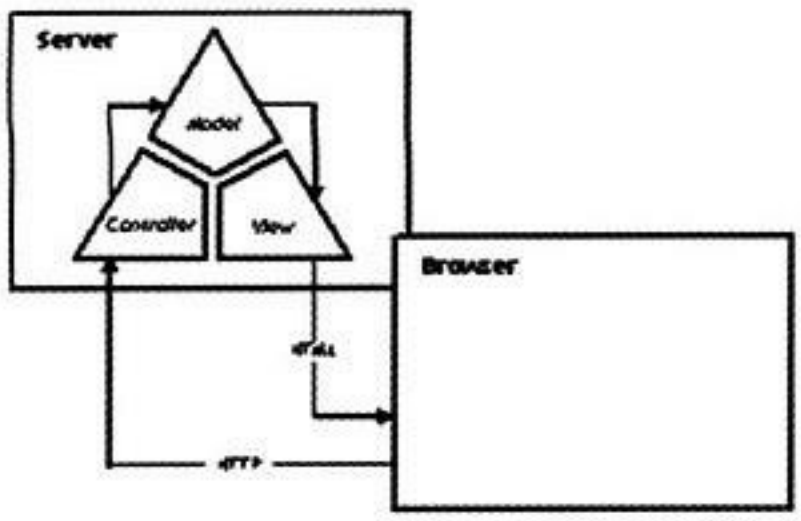


图 7 传统的 mvc 技术

JSON 数据表示技术和 Ajax 技术的结合能很好地解决这一问题，能够实现更彻底的 MVC 架构。

采用 JSON 数据表示和 Ajax 结合的技术后，用户界面视图由纯粹的静态 html 组成，在用户初次请求时由服务器发送到客户端浏览器。此后，用户界面视图在不发生大的变化的情况下，不会有整个页面的刷新，服务器端不再处理视图渲染，视图在客户端浏览器缓存。这时，服务器端与客户端通信数据只有用户提交的数据和服务器产生的 JSON 模型数据，很大程度上减轻了服务器和网络的压力（如图 8 所示）。当然，这部分压力很大程度上追加到了客户浏览器这一端。

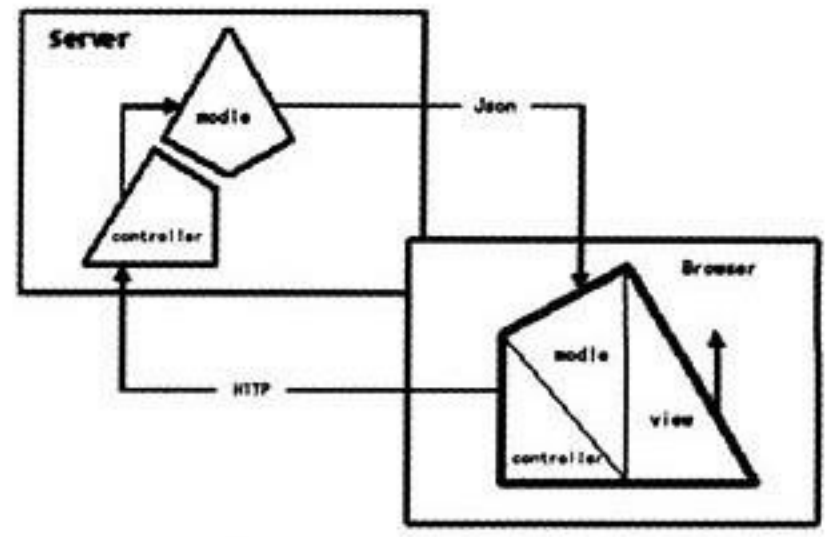


图 8 更彻底的 mvc 架构

6 结语

应用的 Spring+JQuery 技术确实是在工作实践中提炼出来的、在实际的工程实施过程中得到检验的行之有效的方法，期望这套方法也能工作提供一种可行性选择。

（收稿日期：2012-07-27）

微软任命新任大中华区企业传播副总裁

近日，微软大中华区今天宣布庄海欧先生将加入高层领导团队，担任微软大中华区企业传播副总裁，并直接向微软大中华区董事长兼首席执行官贺乐赋报告。

贺乐赋在谈到庄海欧的任命时表示："我很高兴庄海欧先生在这样一个关键时刻加入我们，担任企业传播团队的主管。这段时间对微软来说意义非同寻常，我们向消费者推出了众多创新产品，这在微软的历史上是第一次。同时，我们正努力实现对中国的承诺，不断加强合作，携手中国共创更具创新力、竞争力和人才可持续发展的未来。庄海欧先生丰富的行业经验以及追求卓越的专业精神都将为微软在中国的发展创造更多的价值，发挥重要作用。"

在加入英伟达之前，庄海欧先生在英特尔工作了 13 年，担任多个高级管理职位，包括企业传播总监，领导公司的公共关系和公共事务；大中华区市场总监，负责英特尔在中国大陆和香港的广告、推广、公关、渠道、战略联合营销以及 Intel Inside 计划；英特尔中国区总裁技术助理，同时领导公司与政府事务部门。

庄海欧先生表示："微软是人类历史极具影响力的公司之一，为改善和提高人们的生活质量做出了重大贡献。微软改变了我们沟通、工作、获取信息、娱乐和社交的方式，帮助我们打造无边界的互动社区。我非常高兴能够加入微软，并希望为微软在中国的发展做出积极的贡献。"

利用 Oracle AQ 实现数据异步操作

张扬嵩

摘　要：一般同步应用场景情况下，程序向 Oracle 数据库服务器发送保存或更新数据到 Oracle 数据库表，Oracle 数据库服务器会返回结果。下面讲述利用 Oracle AQ 的功能和异步机制，实现异步更新或保存数据到 Oracle 数据库表中的技巧。

关键词：Oracle AQ 机制；同步；异步

1 引言

一般情况下，程序向 Oracle 数据库服务器发送保存或更新数据到 Oracle 数据库表，Oracle 数据库服务器会返回结果，告诉程序处理成功或失败的结果，以便让程序进行相应的再处理，这是常见的同步的应用场景。但是在有些应用场景，程序不需要实时知道或处理 Oracle 数据库服务器返回的结果，如系统的日志登记，接口的数据传递；为了提高系统的性能、降低系统的耦合，利用 Oracle AQ 的功能，并利用其异步机制，可以简单方便地实现这些功能。

2 实现步骤

程序与 Oracle AQ 交互的方式和接口有多种，这里以实际工作中的应用作为参照并进行举例说明。首先要以 Oracle 系统管理员的角色先创建 Oracle 数据库用户并授操作 Oracle AQ 的相关权限，SQL 代码参考如下：

```
create user aqdev identified by aqdev;
grant connect to aqdev;
grant resource to aqdev;
GRANT EXECUTE ON DBMS_AQ TO aqdev;
GRANT EXECUTE ON DBMS_AQADM TO aqdev;
BEGIN
        DBMS_AQADM.GRANT_SYSTEM_PRIVILEGE
('ENQUEUE_ANY', 'aqdev', true);
DBMS_AQADM.GRANT_SYSTEM_PRIVILEGE
('DEQUEUE_ANY', 'aqdev', true);
END;
```

其次，以新创建的用户 aqdev 登录 Oracle 数据库，创建该用户的队列数据类型，队列表和队列并启动队列，再编写入队列的 Oracle 数据库存储过程，这个存储过程给系统的程序调用，如果是 Java 的程序只要通过 JDBC 访问这个存储过程来间接的操作 Oracle AQ 而不需要通过 Oracle AQ 的 Java API 或 Java JMS 来达到简化程序的目的；并且这个存储过程可以给多种程序设计语言调用，达到比较好的程序复用。代码参考如下：

```
--创建队列数据类型
CREATE OR REPLACE TYPE T_DEV_QUEUE AS OBJECT(
  EXEC_DATE    TIMESTAMP(6),
  PARAM        VARCHAR2(512),
  FLAG         CHAR(1)
);
-- 创建 数据库表
create table AQ_CALLBACK_LOG
(
  LOG_TIME date,
  Log_Msg  varchar2(512),
  LOG_FLAG varchar2(3)
);
--创建队列表
BEGIN
  DBMS_AQADM.CREATE_QUEUE_TABLE(
      queue_table => 'DEV_AQ_TAB',
      multiple_consumers => TRUE,
      queue_payload_type => 'T_DEV_QUEUE');
END;
--创建队列
BEGIN
  DBMS_AQADM.CREATE_QUEUE(
     queue_name  => 'DEV_AQ',
     queue_table => 'DEV_AQ_TAB');
END;
--启动队列
BEGIN
DBMS_AQADM.START_QUEUE(queue_name => 'DEV_AQ');
END;
--入队列的 Oracle 存储过程
create or replace procedure en_msg2aq (p_param in
varchar2,p_flag in varchar2)
IS
    p_payload t_dev_queue;
    enqueue_options    dbms_aq.enqueue_options_t;
    message_properties dbms_aq.message_properties_t;
    message_handle     RAW(16);
    recipients  DBMS_AQ.aq$_recipient_list_t;
```

```
BEGIN
      p_payload := t_dev_queue(sysdate, p_param, p_flag);
recipients(1) := sys.aq$_agent('recver', 'DEV_AQ', NULL);
      message_properties.recipient_list := recipients;
      message_properties.priority := -5;
      message_properties.delay := dbms_aq.no_delay;
      message_properties.expiration := dbms_aq.never;
      enqueue_options.visibility := dbms_aq.on_commit;
      enqueue_options.sequence_deviation := null;
      dbms_aq.enqueue(queue_name   => 'DEV_AQ',
    enqueue_options    => enqueue_options,
     message_properties => message_properties,
                  payload   => p_payload,
       msgid             => message_handle);
  COMMIT;
END;
```

然后编写并运行注册监听程序的代码，实现对指定的队列用指定的存储过程监听，当有数据进入队列时，自动调用注册的监听程序，把数据从队列中取出来进行相应的处理，如保存到 Oracle 数据库表或更新 Oracle 数据库表，这样就实现了数据的异步操作功能，注册监听程序的参考代码如下：

```
DECLARE
      reginfo1     sys.aq$_reg_info;
      reginfolist  sys.aq$_reg_info_list;
BEGIN
       -- 注册了 pl/sql 存储过程 notifyDealMsgCB 当有数据进入队列进被通知处理队列数据
      reginfo1 := sys.aq$_reg_info(
     ' DEV_AQ:recver',
   DBMS_AQ.NAMESPACE_AQ,
         'plsql://notifyDealMsgCB?PR=0',
   HEXTORAW('FF')
     );
      -- 创建注册信息列表
      reginfolist := sys.aq$_reg_info_list(reginfo1);
       -- 进行注册
      sys.dbms_aq.register(reginfolist,1);
END;
```

最后就是要实现监听的存储过程来处理队列里的数据，这个监听存储过程是事件驱动的，当队列发生入队事件时，由 Oracle 数据库服务器自动触发调用；参考代码如下：

```
create or replace procedure notifyDealMsgCB (
 context raw,
  reginfo sys.aq$_reg_info,
  descr sys.aq$_descriptor,
 payload raw,
  payloadl number)
IS
 dequeue_options   DBMS_AQ.dequeue_options_t;
 message_properies DBMS_AQ.message_properties_t;
 message_handle    RAW(16);
      message     t_dev_queue;
BEGIN
      dequeue_options.msgid := descr.msg_id;
dequeue_options.consumer_name := descr.consumer_name;
  DBMS_AQ.DEQUEUE(
        queue_name => descr.queue_name,
   dequeue_options => dequeue_options,
      message_properties => message_properies,
                payload => message,
                msgid => message_handle);
  insert into AQ_CALLBACK_LOG (LOG_TIME,Log_Msg,
LOG_FLAG) values(sysdate,message.param, message.flag);
     commit;
END;
```

3 应用说明

入队 en_msg2aq 存储过程可以直接在 PL/SQL 中运行使用；不过一般是给程序调用，Java 的程序可以通过 JDBC 接口调用存储过程；C 相关的程序可以通过 OCI 接口或 PROC 调用存储过程；调用存储过程是件简单的事了。这里举个通过 JDBC 调用的实例，代码参考如下：

```
import java.sql.*;
import java.io.*;
class CallOraSP
  {
    public static void main (String args [])
        throws SQLException, IOException
    { // 加载驱动
      DriverManager.registerDriver (new oracle.jdbc.driver.
OracleDriver());   // 连接数据库
       Connection conn =
          DriverManager.getConnection   ("jdbc:oracle:oci8:
@oralocal", "aqdev", "aqdev");
       CallableStatement cstmt = conn.prepareCall   ("{call
en_msg2aq(?, ?)}");
  cstmt.setString (1, "hello world");  // The name argument is
// the second ?
cstmt.setString (2, "0"); // The raise argument is the third ?
       cstmt.execute ();
       cstmt.close();
  conn.close();   } }
```

4 结语

利用 Oracle AQ，并结合 Oracle 的存储过程来创建 Oracle 数据库数据异步操作的框架降低了系统的接口之间的耦合，提高了系统的性能，并且整体框架比较简单、可维护好、复用性高。

（收稿日期：2012-07-25）

基于 Excel 的职业技能业务管理系统设计与实现

刘仁轩

摘　要：为了保障市级职业技能管理中心业务的管理和维护，设计了基于 Excel 的业务管理系统，并给出了实现的核心代码。

关键词：Excel 表处理；VBA 应用；职业技能

1　引言

长期以来，职业技能管理中心对历年参加职业技能考试的所有考生从报名、审批、核准、培训、考试、复核、发证、向各个相关部门呈报等各个环节进行全面管理。但是，由于缺少强有力的专业管理软件进行电化管理，这样在实际处理中难以避免地出现了大量的错误与混乱，导致业务效率低下、准确性差、回溯性不强等问题。鉴于此，本管理系统的出现成为一种必然。这里所设计的管理系统可以对职业技能管理中心的考生进行全方位、多角度的管理，保证了业务需求、大幅度提高了工作效率。

2　Excel 和 VBA

Excel 是 Windows 环境下应用最广泛的自动化办公软件之一，功能强大、人机界面友好、使用方便易学。其数据处理、统计分析和计算功能非常强大。VBA（Visual Basic for Application）是 Windows 应用软件的通用控制语言，其独特之处在于它由应用程序控制，反过来，它又可以增强应用程序的功能。在 Excel 2003 中，VBA 已经得到了完美的支持。

基于以上原因，软件在开发中充分考虑到实际应用的各个环节，在保留了 Excel 强大的数据处理功能的同时，提供了多个功能模块来满足各阶段的业务需求。

3　系统界面

系统运行后提供了如图 1 所示的菜单。

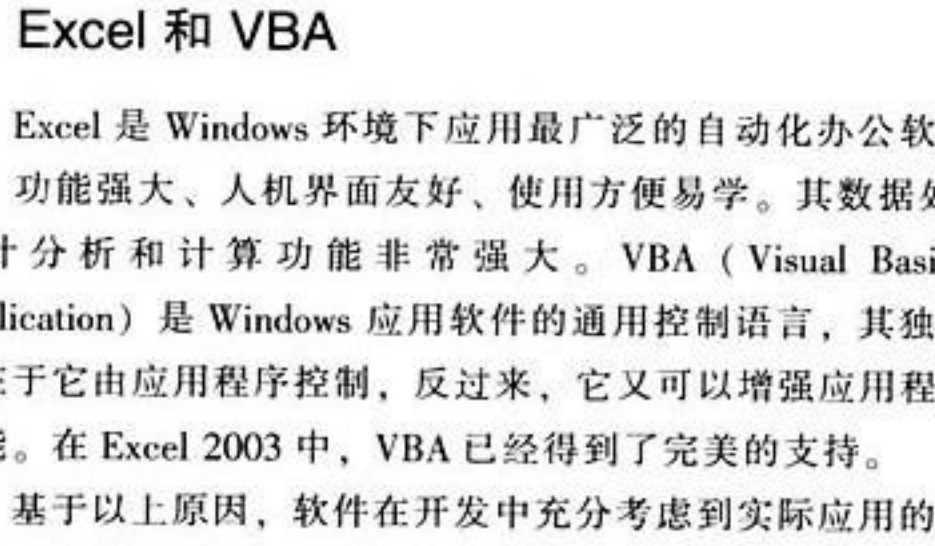


图 1　菜单界面

4　系统分析

由于本管理系统对职业技能管理中心的业务起到了全面的支撑作用，所以系统设计上采用模块化设计方法，整体结构为树形层次结构。基本结构如图 2 所示。

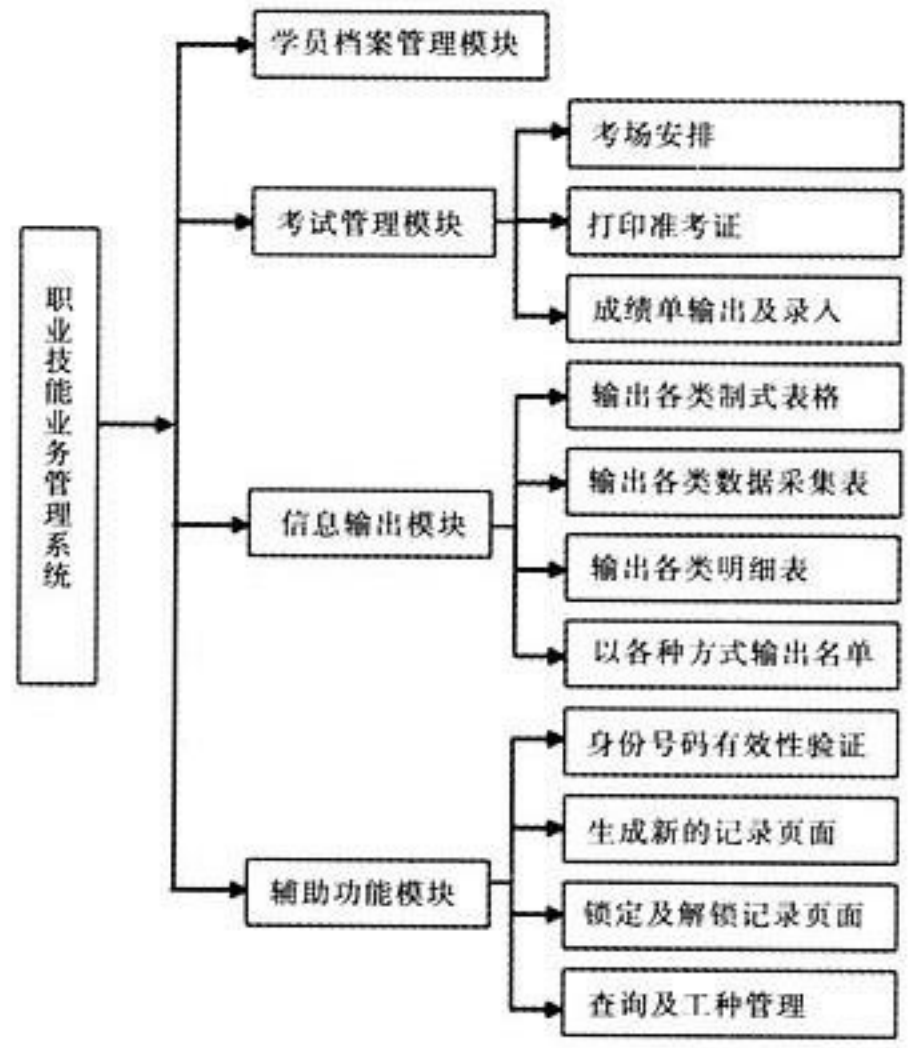


图 2　系统结构

5　系统实现

5.1 数据处理的底层

一份 Excel 文档可以看做一个 Excel 数据库，而其中的一个工作簿是一个 Excel 数据表。系统将一期培训学员的档案放置在一个工作簿中，为了编码方便，设计中提出如下类 ConnectionEx，专门用于提供对特定工作簿的 ADO.Connection 的访问对象：

```
Class ConnectionEx
Private m_cn As New ADODB.Connection
```

```
Private Sub Class_Initialize()
    On Error Resume Next
    Dim oBook As Workbook
      Set oBook = Application.ActiveWorkbook
      ' 初始化
      m_cn.Open   "Provider  =Microsoft.Jet.OLEDB.4.0;Data
Source=" & oBook.Path & "\" & oBook.Name & ";Extended
Properties=Excel 8.0"
    If Err.Number <> 0 Then
        MsgBox Err.Description, , "出现错误"
    End If
End Sub
Private Sub Class_Terminate()
    m_cn.Close
End Sub
Public Function RunSQL(sql As String) As Recordset
    Dim rs As New ADODB.Recordset
    rs.Open sql, m_cn, adOpenDynamic, adLockOptimistic
    Set RunSQL = rs
End Function
End Class
```

5.2 明细表输出

由于在实际应用中需要输出多种形式的明细表，为此系统提供了完整的明细表输出功能，用户可以根据自己的需要选择输出。执行该功能后出现如图 3 所示表输出界面。

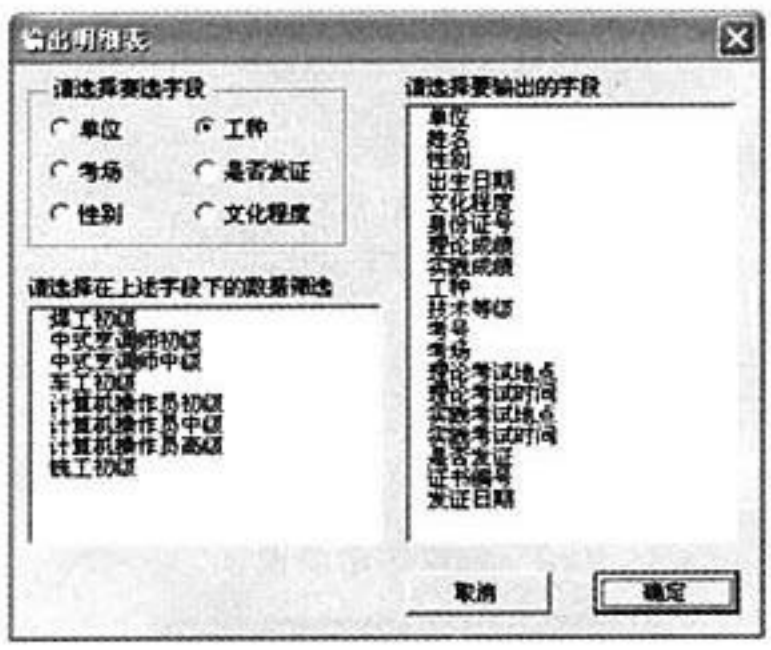


图 3 输出界面

明细表输出模块核心代码如下：

```
Public Sub doo()
    Dim osheetSrc As Worksheet
    Dim f As New FormExportList
    Set osheetSrc = Application.ActiveSheet
    f.Init osheetSrc
    f.Show
    If f.m_OK = False Then Exit Sub
    Dim oSheetTarget As Worksheet
    osheetSrc.Copy
     Set oSheetTarget = Application.Workbooks (Application.
Workbooks.count).Worksheets(osheetSrc.Name)
    oSheetTarget.Range("A2").Select
    oSheetTarget.Activate
    ActiveWindow.FreezePanes = False
    ActiveWindow.Split = False
    Dim rowIndex As Integer
    Dim c 字段 1 As Integer
    Dim c 字段 2 As Integer
    Dim sz As String
    c 字段 2 = -1
    If f.m_sz 当前过滤字段 <> "工种技术等级" Then
        c 字段 1 = ColumnName2ColumnIndex(oSheetTarget,
f.m_sz 当前过滤字段)
    Else
        c 字段 1 = ColumnName2ColumnIndex(oSheetTarget, "
工种")
        c 字段 2 = ColumnName2ColumnIndex(oSheetTarget, "
技术等级")
    End If
    rowIndex =MAX_ROWCOUNT
    Do
        sz = "" & oSheetTarget.Cells(rowIndex, c 字段 1)
        If c 字段 2 <> -1 Then sz = sz & oSheetTarget.Cells
(rowIndex, c 字段 2)
        If sz <> "" Then
            If f.值是否可以输出(sz) = False Then
                oSheetTarget.Rows(rowIndex).Delete
            End If
        End If
        rowIndex = rowIndex - 1
    Loop Until rowIndex = 1

    Dim colIndex As Integer
    colIndex = MAX_COLUMNCOUNT
    Do
        sz = oSheetTarget.Cells(1, colIndex)
        If sz <> "" Then
            oSheetTarget.Columns(colIndex).Delete
        End If
        colIndex = colIndex - 1
    Loop Until colIndex = 1
End Sub
```

6 结语

系统在开发中充分考虑到实际应用的各个环节，在保留了 Excel 强大的数据处理功能的同时，提供了多个功能模块来满足各阶段的业务需求。由于本系统是在紧跟实际业务的过程中开发而成，所以能紧贴业务，最大限度地满足实际需求，大幅提高了工作效率。

（收稿日期：2012-09-07）

基于 C 和 C++ 的无线打分器数据传输程序设计

卞晓强　边晓明

摘　要：给出了基于 STC89C52 和 NRF905 模块的无线打分器和基于串口通信上位机软件的程序设计方法。用 C 语言和 C++语言，对数据传输处理的过程进行了剖析。

关键词：输入数据；数据编号；分时发送；数据接收；串口接收；数据归类

数据的无线传输技术越来越受到消费者欢迎，并且有着广阔的发展前景，逐渐成为生活中必不可缺少的一项实用技术。文中尝试着制作了一套无线远程打分器，本作品基于无线传输数据技术设计，它是以 STC89C52 单片机为控制核心，以 nrf905 无线传输模块来实现数据的生成和无线发送和接收（如图 1 所示）。本制作的控制核心选用了基础的 51 单片机，它成本低，易于使用和操作，完全适用于日常生活中。

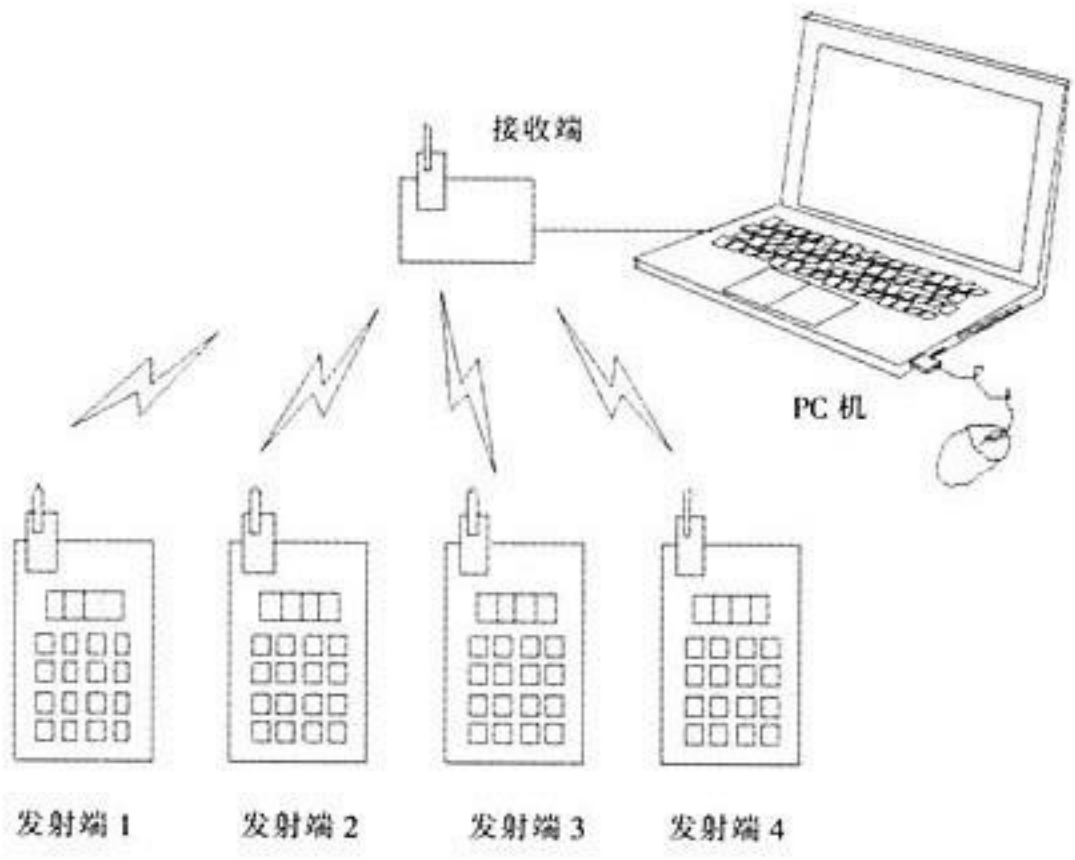


图 1　传输示意图

1　发送端组成

电路原理图如图 2 所示。

实物如图 3 所示。

(1)　控制：STC89C52 单片机。

(2)　无线发送：NRF905。

(3)　数据输入：ZLG7290 和按键。

(4)　数据显示：数码管。

(5)　其他：蜂鸣器。

(6)　供电：9v 变压器、LM7805 和 ASM117。

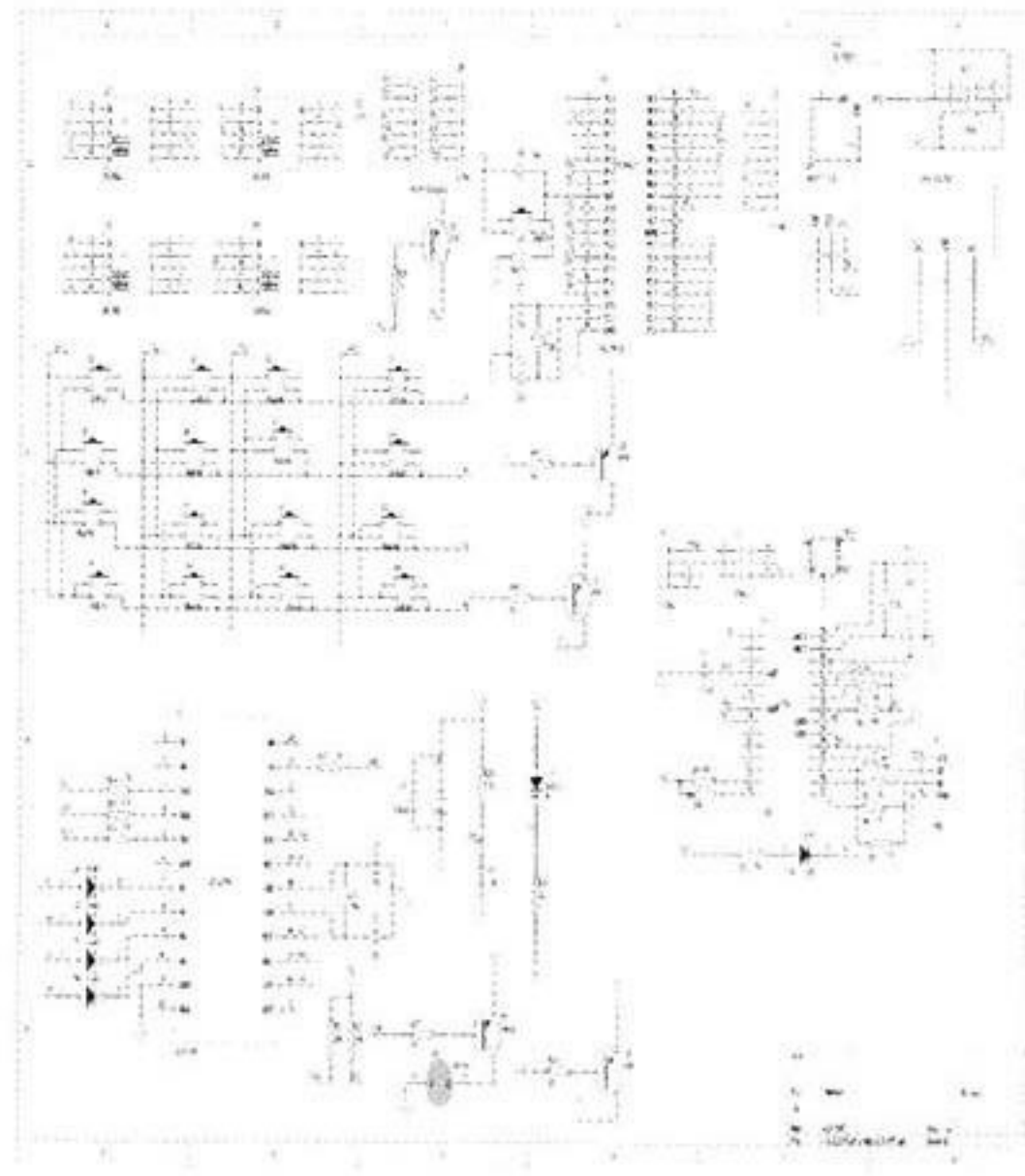


图 2　发送端电路原理

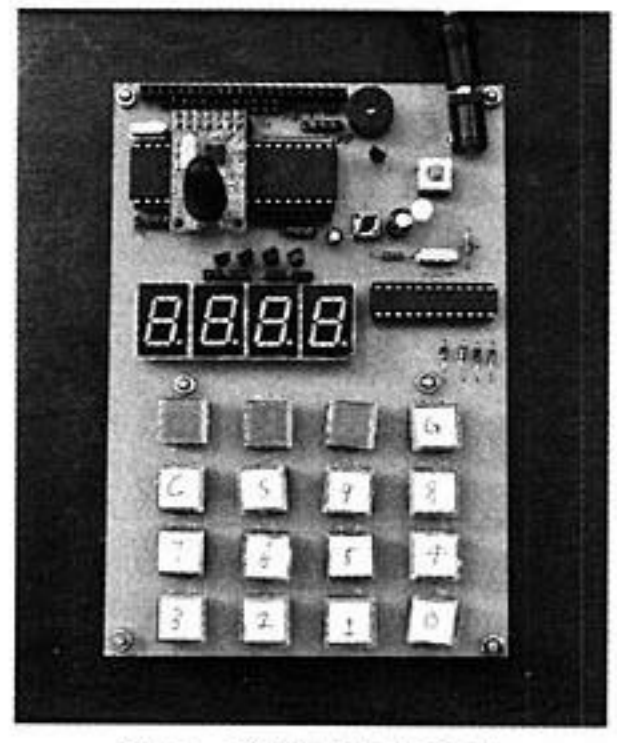


图 3　发送端实物设计

2 接收端组成

电路原理图如图 4 所示。

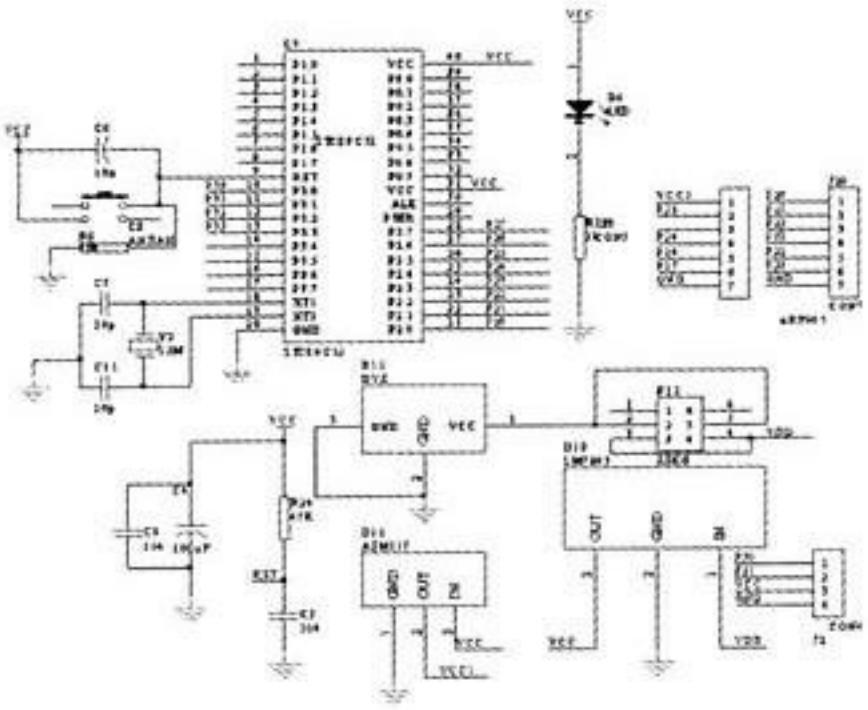


图 4 接收端电路原理

实物如图 5 所示。

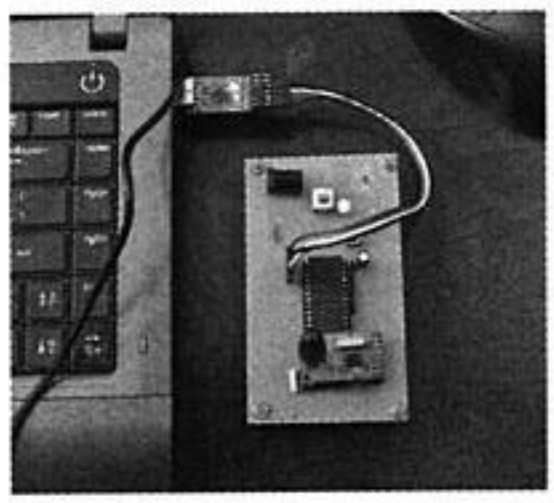


图 5 接收端实物设计

(1) 控制：STC89C52 单片机。

(2) 无线接收：NRF905。

(3) 串口通信：RS232-USB 接口转换器。

(4) 供电：9v 变压器、LM7805 和 ASM117。

3 上位机软件

运行如图 6 和图 7 所示。

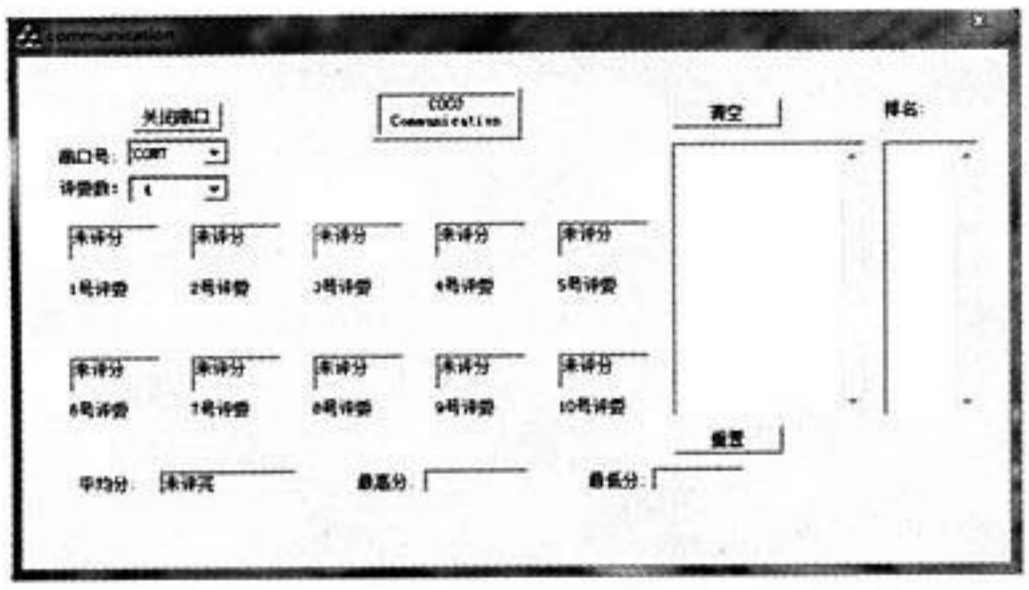


图 6 软件界面设计

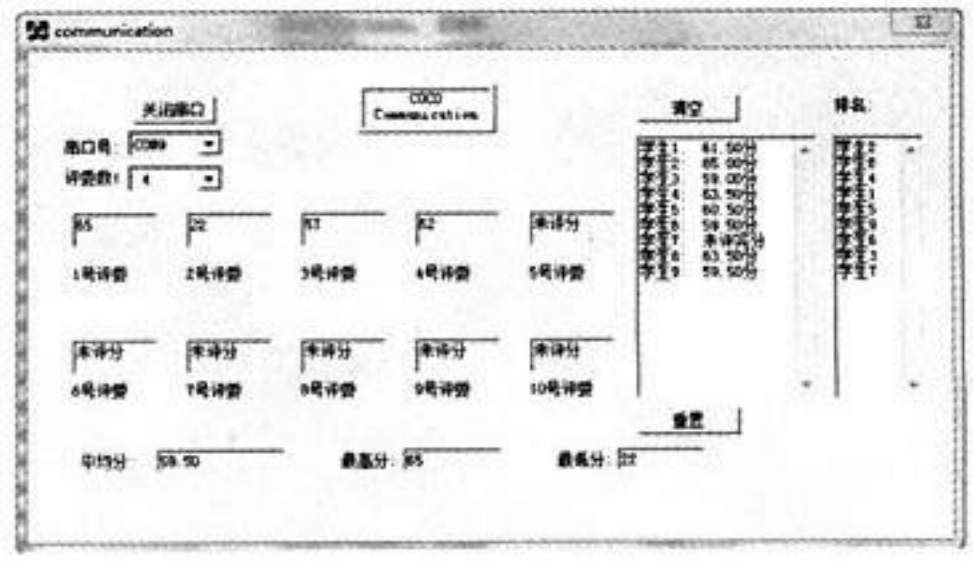


图 7 软件运行效果

4 编程思路

4.1 输入数据

数据的输入通过键盘、蜂鸣器和数码管完成，每摁一个键就会给用户一个视觉或听觉上的反馈。

为了防止非法的数据输入，必须给输入定一个流程：(1) 输入开始确定；(2) 输入过程；(3) 输入完成确认；(4) 输入值判定。

```
    Unsigned char stop1=0,start1=0;
//stop1 为输入过程中的判定值,start1 为输入开始的判定值
    char severe=-1;
//severe 为摁键返回值
    while(1)
      {
        severe=key1();
//首先读取键值
     if(severe! =-1)
//-1 表示没有键摁下
     {
        if(severe==10){start1=1;sm[3]=8;}
//开始确认键按下后才可进行打分,键值 10 为开始确认键
        if (severe >=0&&severe <=9&&start1 ==
1&&stop1>=0&&stop1<=2)
//stop=0 时表示没有输入值,stop=n 时表示没输入值的位数
        {
        switch(stop1)
        {
         case 0:sm[0]=severe;stop1=1;break;
         case 1:sm[1]=severe;stop1=2;break;
         case 2:sm[2]=severe;stop1=3;break;
         default:break;
         }
//sm 为数码管的显示值,每摁下个有效按键,数码管都会更新显
//示
        }

        if(severe==11&&start1==1&&stop1>=0)
//键值 11 是清除键,表示清空当前分数
```

```
        {stop1=0;sm[0]=10;sm[1]=10;sm[2]=10;}
//显示清空,输入值位数为 0
        if(severe==12&&start1==1&&stop1>=1)
//键值 11 是输入完成确认
        {
        if(stop1==1)score=sm[0];
        if(stop1==2)score=sm[0]*10+sm[1];
            if (stop1==3){if (sm [0]==1){score=sm [0]*100+
10*sm[1]+sm[2];}else score=101;};
//对输入数据做运算处理得出所输入的分数大小
if(score>=0&&score<=100)
//判断分数是否合法
{
start1=0;
stop1=0;
sm[3]=10;sm[0]=10;sm[2]=10;sm[1]=10;
speak (100);delay (2000);speak (100);delay (2000);speak (100);
delay(2000);
}
//发送成功,是蜂鸣器发出 3 段清脆的铃声,然后初始化输入状
//态
Else
{
sm[0]=8;
sm[2]=8;
sm[1]=8;
speak(1800);
//输入不合法,数码管显示 3 个 8 和蜂鸣器发出一段刺耳的声音
//来给用户一个提醒
    }
        }
    }
        Display(sm);
//数码管显示
}
```

4.2 数据编号

为区别不同的发送端，要给几个发送端都进行一个特有的编号，以便进行识别和判定。所以在将数据发送之前必须给各个数据一个编号：

```
#define bianhao  1
if(score>=0&&score<=100)
 nRF905_TxRxBuf[3] =bianhao;
//nRF905_TxRxBuf 为存储发送数据的数组
```

4.3 分时发送

传统意义上的 nrf905 无线模块通信只能以点对点的形式进行，当多台 nrf905 发射模块向一台 nrf905 接收模块传递信息时，会发生信号丢失的情况。为避免在打分过程中丢失信息，采取将一个数据按照不同的频率重复发送 1000 次，总发射时间小于 1S，不同频率使不同单片机信号的发射产生间隔。经试验 100 次，没有发现丢失掉一次信息。以这种方式完成多对点的信号通信，采用分时信号传送的办法，形成信号的错时传送，在民用系统通信中是可行的：

```
    #define  fre   800
    //控制发送频率
    int s=0;
    s++;
    //计数
    if(s==fre)
    {
    nRF905_Tx();
    //发送数据
    s=0;
    }
```

4.4 数据接收

在数据接收端主要是要将接收到的数据进行分类并通过串口发送，该程序用 51 单片机的定时器来实现波特率的产生：

```
int score[10]={0};
int m[10]={0,0,0,0,0,0,0,0,0,0};
int h=0;
  while(1)
    {
        int i;
        SCON  = 0x50;
/* SCON: 模式 1, 8-bit UART, 使能接收          */
        TMOD |= 0x20;
/* TMOD: timer 1, mode 2, 8-bit reload         */
        TH1   = 0xFD;
/* TH1:  reload value for 9600 baud @ 11.0592MHz   */
        TR1   = 1;
/* TR1:  timer 1 run                           */
        EA    = 1;
/* 打开总中断 */
        //ES    = 1;
TI    = 1;


    nRF905_Init();
    nRF905_Config();
    delay(500);
//系统初始化
    h++;
    nRF905_Rx();
                sm[3]=nRF905_TxRxBuf[0];
                sm[2]=nRF905_TxRxBuf[1];
                sm[1]=nRF905_TxRxBuf[2];
                sm[0]=nRF905_TxRxBuf[3];
//nrf05 接收数据
    score[sm[0]]=sm[0]*1000+sm[1]*100+sm[2]*10+sm[3];
//数据处理和运算
```

```
  m[sm[0]]=1;
//将已发出数据的发送端标记
            if(h==5)
   {
   for(i=0;i<10;i++)
   {
    if(m[i]&&(score[i]%10))
   {if(i==0){printf("00%daaa",score[i]);}
    else printf("%daaa",score[i]);
//通过串口传输数据,a 为数据间隔的标志
    }
   } h=0;
    }
 }
```

4.5 串口接收

串口上位机程序是通过 VC6.0 上的 MFC 完成的，主要用 MSCOMM 控件来实现数据的接收：

```
CDialog::OnInitDialog();
if(!  m_comm1.GetPortOpen())
//判断串口是否已经打开
{
m_comm1.SetCommPort(7);
//选择串口号
m_comm1.SetPortOpen(TRUE);
//打开串口
m_comm1.SetRThreshold(2);
//收到两个字节引发 OnComm 事件
m_comm1.SetInputMode(1);
//输入模式选为二进制
m_comm1.SetSettings("9600,n,8,1");
 //设置串口参数,波特率 57600,无奇偶校验,1 位停止位,8 位
//数据位
MessageBox("串口初始化完毕","提示");
//提示串口成功初始化
}
else MessageBox("串口被占用","提示");
 //如果已经打开串口,消息框提醒
m_serial.SetWindowText("关闭串口");
//按钮显示状态改变
return TRUE;
```

4.6 数据归类

4.7 在接收到数据后将数据归类和显示

```
VARIANT variant1;
//定义 VARIANT 型变量,用于存放接收到的数据
COleSafeArray safearray;
//定义 safearray 型变量
LONG len,k;
//定义长整型变量 len,k
BYTE rxdata[2048];
//定义 BYTE 型数组
CString stremp1,stremp2;
//定义两个字符串
int s=0,x=0,i,max=0,min=100;
if(m_comm1.GetCommEvent()==2)
 //判断引起 OnComm 时间的原因
{
//如果是接收到特定个字节数,则读取接收到的数据
variant1 = m_comm1.GetInput();
//把接收到的数据存放到 VARIANT 型变量里
safearray = variant1;
//VARIANT 型变量转换为 ColeSafeArray 型变量
len = safearray.GetOneDimSize();
for(k=0;k<len;k++)
{
safearray.GetElement(&k,rxdata+k);
 //得到接收到的数据,放到 BYTE 型数组 rxdata 里
}
for(k=0;k<len;k++)
{
BYTE bt = (*( char*)(rxdata+k));
if(bt==97){stremp1=strRXDdata;strRXDdata="";}
else  strRXDdata+=bt;
 //将字符送入临时变量 strtemp 存放
}
if(stremp1[0]=='0')
{stremp2=stremp1;m  [0]=1;score  [0]=atoi(stremp2);stremp2.
Format("%d",score[0]);SetDlgItemText(IDC_EDIT2,stremp2);}
else if(stremp1[0]=='1')
{stremp2=stremp1;m  [1]=1;score  [1]=atoi  (stremp2)-1000;
stremp2.Format  ("%d",score  [1]);SetDlgItemText(IDC_EDIT4,
stremp2);}
else if(stremp1[0]=='2')
{stremp2 =stremp1;stremp2.Delete (0);m [2] =1;score [2] =atoi
(stremp2);if  (stremp2  [0]  =='0')  {stremp2.Delete  (0);}
SetDlgItemText(IDC_EDIT3,stremp2);}
else if(stremp1[0]=='3')
{stremp2 =stremp1;stremp2.Delete (0);m [3] =1;score [3] =atoi
(stremp2);if  (stremp2  [0]  =='0')  {stremp2.Delete  (0);}
SetDlgItemText(IDC_EDIT1,stremp2);}
else if(stremp1[0]=='4')
{stremp2 =stremp1;stremp2.Delete (0);m [4] =1;score [4] =atoi
(stremp2);if  (stremp2  [0]  =='0')  {stremp2.Delete  (0);}
SetDlgItemText(IDC_EDIT5,stremp2);}
else if(stremp1[0]=='5')
{stremp2 =stremp1;stremp2.Delete (0);m [5] =1;score [5] =atoi
(stremp2);if  (stremp2  [0]  =='0')  {stremp2.Delete  (0);}
SetDlgItemText(IDC_EDIT6,stremp2);}
else if(stremp1[0]=='6')
{stremp2 =stremp1;stremp2.Delete (0);m [6] =1;score [6] =atoi
(stremp2);if  (stremp2  [0]  =='0')  {stremp2.Delete  (0);}
SetDlgItemText(IDC_EDIT7,stremp2);}
```

(下转第 59 页)

用 C# 定制实用的 FTP 工具

徐怀平

摘　要：FTP 的应用非常广泛，在业务中对 FTP 的使用比较频繁，如何更便捷快速地更新，是一件非常重要的事，有时在处理工具及程序发布时，是否发布正确，进行本地版本与 FTP 版本的对比同样也很有价值，文中介绍如何实现这一专业定制的 FTP 工具，使工程和业务维护时更为方便。

关键词：FTP 工具；C# 语言；更新；同步；比对

1　实现目标与思路

在业务系统中，软件的发布、内部工具的更新等，FTP 都起着极为重要的作用。

介绍如何使用 C# 编程实现一个专用的 FTP 工具，用于完成业务系统的上传、更新、同步等功能，以及提供了业务中 FTP 对比这样更具有特色功能的工具。本工具所具有的功能特色是维护 FTP 人员最实用功能的快速体现。

1.1　目标

常用的 FTP 工具大多基于交互界面，为了工具使用的方便性，采用命令行的方式来实现对应的功能更有价值，因此，需要达成以下目的：

（1）用命令行完成一个目录的上传下载（非常适用于更新，能够双击或者是一个快捷键就完成了 FTP 的同步）。

（2）FTP 是否传输正确，通过命令行完成比对（提供快速比对和完全比对），它实际上等于全部下载后（在内存）比较；也可进行快速比对，每个文件首先判断大小是否相同，大小相同，只取指定长度的内容比较，起初步判断作用。

因此对工具的使用定义成以下的格式：

（1）MyFtp –up d:\sj Ftp 目录：完成 d:\sj 上传到 FTP 目录。

（2）Myftp –down FTP 目录 d:\sj：完成从 FTP 目录下载到 d:\sj 目录。

（3）Myftp –c d:\sj FTP 目录：完成本地目录 d:\sj 是否都已经正确上传至 FTP 的确认。

（4）Myftp –c –q5120 d:\sj FTP 目录：功能和上面相同，但是–q 的选项用于快速比较，–q 后面表示比较的字节数，如果文件长度相同的前提下，读取相应的字节数和本地文件进行比较是否相同，完成一定程度上的快速比较功能。

1.2　基本思路

为了使工具具有方便性，因此要实现：

（1）根据传入的命令行参数解析对应功能。

（2）针对不同目标，完成对应功能处理。

1.3　难点

其实仔细分析一下 Ftp 的特点，要实现对应功能，将面临几个基本问题：

（1）FTP 文件的上下传处理：

在上下传片理中，使用系统的 API 方式处理 FTP 的方法，因此需要集中的一个 namespace 中来处理。

（2）FTP 目录单进程非递归遍历的实现方法

由于与 FTP 交互是一个单线程的过程，但是 FTP 目录和本地文件目录结果类似，它是一个树型结构，如何处理它的单线程又无法使用递归实现的问题，在程序中使用了数组来实现它的递归效果；并且在每次处理时，实现只处理目录的当前层，勿需同时在多个目录间来回折腾。

（3）虚拟动态数组

由于使用的目录和文件数的总个数并不确定，因此在程序中要实现数组的动态扩展功能，其实只这只要使用 C# 中的 list 类型就可以实现，这样容器最终可以保留下来所有的子目录和文件列表。

2　具体实现

根据功能划分，主流程是命令行解析、对应功能的处理。下面就按命令行解析和具体每一项功能的处理来进行分析和说明。下面针对主程序和各功能的实现进行描述。

2.1　命令行的解析处理

在 C# 的主程序处理中和 C 非常类似，对于传入的参数在 main 主程序中（program.cs）：

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using System.IO;
namespace MyFtp
{
```

```
static class Program
{
    [STAThread]
    static void Main(string[] args)
    {
        Form_MYFTP  RunForm;
        string      ls_size;
        if (args.Length == 0){
            Application.Run(new Form_Tips());
            return;
    }
    if ( args.Length<3 ){
        Console.WriteLine("参数不足");
        return;
    }
    RunForm = new Form_MYFTP();
    RunForm.Show();
    switch (args[0].Trim().ToLower()){
         case "-up":    //上传至 FTP
            RunForm.fun_ftpUp(  args[1], args[2] );
            break;
        case "-down":   //从 FTP 下载
            RunForm.fun_ftpDown( args[2], args[1] );
            break;
        case "-c":      //本地目录与 FTP 上进行比较
            ls_size = args[1].Trim().ToLower();
            if ( ls_size.Substring(0,2) == "-q" ){
                ls_size = ls_size.Substring( 2 );
                RunForm.fun_ftpCompare (args [2],  args [3],
Int16.Parse(ls_size));
            }else{
                ls_size = "0";
            RunForm.fun_ftpCompare( args[1], args[2], 0 );
            }
            break;
        default:
            Console.WriteLine("非正确参数! ");
            return;
    }
    Application.Run(RunForm);
        }
    }
}
```

针对命令行中不同的参数进行分支处理，转向了具体的功能实现，分别由上传、下载、对比，在对比功能中有辅助参数，就是对比时选择长度的功能处理。通过分支处理后，程序就显得条理清晰。

接下来将讲述如何实现 FTP 的上传、下载与对比功能。

2.2 FTP 处理

对于 FTP 的子功能处理，全部 namespace MyProFTP 中，封装了调用系统 wininet.dll 时使用的：

(1) 结构：FILETIME、WIN32_FIND_DATA、Itemftp。

(2) 函数调用类 FtpDll。

对于具体的 FTP 处理使用以下的处理流程：

(1) 首先连接 FTP。

(2) 取得 FTP 上的文件目录清单。

(3) 下载时将 FTP 上的文件同步到本地目录。

(4) 如果是文件比对，则和下载相似，只要将 FTP 上文件内容读取到内容和本地文件内容进行比较（二进制流的比较）。

(5) 如果是快速比较，则从 FTP 上读取的信息只要是用户指定的长度即可。

有关 FTP 具体功能的处理，定义了 namespae myftp，里面有 FTP 上下传与对比的处理方法。

2.2.1 从 FTP 下载

功能使用方法说明：

MyFtp –down his3/his3@192.168.1.100:21/his3 d:\his3

表示从远程的 192.168.1.100 端口为 21，下载目录 his3 下所有内容到本地的 d:\his3 目录下，使用用户 his3，密码为 his3。实际效果如图 1 所示。

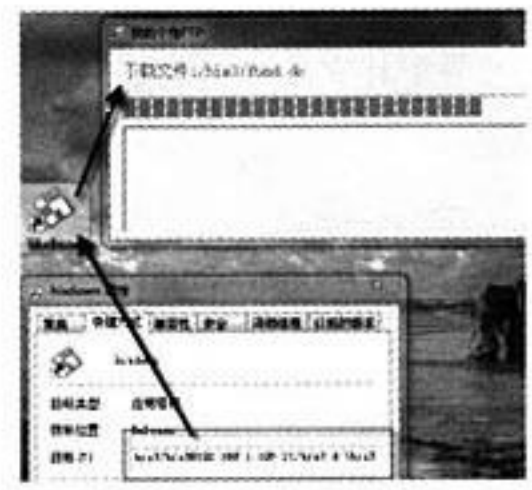


图 1　下载运行界面

可以看出，进行下载时，如果设置好命令行参数，只要一次双击就可以完成，非常方便。下面是处理 FTP 下载的 fun_ftpDown，可以看出过程比较精简：

```
    public Boolean fun_ftpDown(string strLoc, string strFtp){
        int li_which, li_countfiles;
        string ls_Ret;
        FtpDll MyFtp = new FtpDll();
        ls_Ret = MyFtp.fun_Connect( strFtp );
        if (ls_Ret ! = "OK"){
            rtb_info.Text = ls_Ret;
            return false;
        }
        li_countfiles = MyFtp.fun_FtpDirs();
        pb_info.Maximum = li_countfiles;
        pb_info.Step = 1;
 for (li_which = 0; li_which < li_countfiles; li_which++){
rtb_info.Text = "下载文件:"+MyFtp.fun_filename(li_which);
            pb_info.PerformStep();
```

```
            Application.DoEvents();
            MyFtp.fun_downfile(li_which, "/9401", strLoc );
        }
        return true;
```

在下载过程中：

（1）通过 MyFtp.fun_Connect 连接指定的 ftp 信息：在 fun_Connect 中完成对 FTP 命令行参数中指定的 FTP 进行连接，通过 InternetOpen 来初始化连接，并使用 InternetConnect 来完成使用指定用户进行建立链接。

（2）MyFtp.fun_FtpDirs 完成了对 FTP 目录及子目录的遍历：实际上在类 ftpdll 的私有函数 private int fun_FtpDirs (string as_dir, Boolean ab_Build) 中，把当前目录下的目录名与文件名加入到列表中。因此使用类型 List<Itemftp> 来作为动态的文件列表，保留了文件名、FTP 目录、全路径文件名和是否目录 4 个属性，在每一次调用了函数后，列表就会自动变化，这样就通过循环的方式完成了对递归方式的实现，由于 FTP 是单线程连接，因此只有一个当前目录，如果要不同目录访问需要同时每次切换，这样就会造成较大的开销。因此每次访问一个目录的当前层时，处理完当前层的的目录与文件是比较好的方法。

下面是 fun_FtpDirs 的具体内容，它完成了 Ftp 目录的遍历，实际上它是通过单循环完成的，只是目标数目是一个不断增长直至遍历完成为止：

```
public int fun_FtpDirs(){
    int li,li_files;
    li_files = fun_FtpDirs(is_ftpRoot, true);
    for (li = 0; li < li_files; li++){
      if (ftpfiles[li].bDir ){
          if (ftpfiles[li].cFile == "." || ftpfiles[li].cFile == "..")
              ftpfiles[li].cFile = "";
          else{
 fun_FtpDirs(ftpfiles[li].cPath + "/" + ftpfiles[li].cFile, false);
              li_files = ftpfiles.Count;
          }
      }
    }
    return li_files;
}
```

（3）根据取得所有文件，依次下载在 FTP 上的所有文件，使用 MyFtp.fun_downfile 来完成：此功能只是对指定的文件进行下载。

由于工具通过系统级的 wininet.dll 来对 FTP 进行处理，因此功能上还具有非常强的兼容性。为保证文件的正确性，文件传输时使用流格式即 BINARY。

2.2.2 上传至 FTP

功能使用方法说明：

MyFtp - up d:\his3 his3/his3@192.168.1.100:21/his12

表示从本地 d:\his3 目录上传至远程 FTP 上的 his12 目录，实际效果如图 2 所示。

图 2 上传运行界面

可见整个上传操作也同样是非常简单，在文件上传的过程来说，同样也比较简洁：

```
public Boolean fun_ftpUp(string strLoc, string strFtp){
    int li_which, li_countfiles=1;
    FtpDll MyFtp = new FtpDll();
    if (! fun_locfiles(strLoc))
        return false;
    MyFtp.fun_Connect(strFtp);
    li_countfiles = locfiles.Count;
    pb_info.Maximum = li_countfiles;
    pb_info.Step = 1;
    for (li_which = 0; li_which < li_countfiles; li_which++){
        rtb_info.Text = "上传文件:" + locfiles[li_which].cFile;
        pb_info.PerformStep();
        Application.DoEvents();
          MyFtp.fun_upfile (locfiles [li_which].cFile, locfiles
[li_which].bDir, strLoc);
    }
    return true;
}
```

主要是两个主流程：

（1）取得本地的所有文件与文件夹的清单，通过函数 fun_locfiles 来完成：

```
private Boolean fun_locfiles(string as_filepath){
    int i, j, k;
    int li_countdir;    //全部的目录数
    string[] ls_dirs = new string[1000];     //所有目录,目前仅预
//定义个
    string[] ls_subdirs;    //用于取得子目录下的所有目录
    string[] ls_subfiles;                  //当前目录下面的文件数
ItemFile MyItem;
if (! System.IO.Directory.Exists(as_filepath)){
    rtb_info.Text = as_filepath + " 目录不存在! ";
    return false;
}
MyItem = new ItemFile();
locfiles.Clear();
MyItem.bDir = true;
MyItem.cPath = as_filepath;
MyItem.cFile = as_filepath;
locfiles.Add(MyItem);
i = -1;
```

```
    li_countdir = 1;
    do{
        i++;
        if (! locfiles[i].bDir)
            continue;
        ls_subdirs = System.IO.Directory.GetDirectories
(locfiles[i].cPath);
        li_countdir = ls_subdirs.Length;
        if (li_countdir > 0){
            for (j = 0; j < li_countdir; j++){
                MyItem = new ItemFile();
                MyItem.cPath = ls_subdirs[j];
                MyItem.cFile = ls_subdirs[j];
                MyItem.bDir = true;
                locfiles.Add(MyItem);
            }
            ls_subdirs = null;
        }
        //搜索当前目录所有文件
        ls_subfiles = System.IO.Directory.GetFiles (locfiles[i].
cPath, "*.*");
        if (ls_subfiles.Length > 0){
            k = 0;
            for (k = 0; k < ls_subfiles.Length; k++){
                MyItem = new ItemFile();
                MyItem.bDir = false;
                MyItem.cPath = locfiles[i].cPath;
                MyItem.cFile = ls_subfiles[k];
                locfiles.Add(MyItem);
            }
            ls_subfiles = null;
        }
    }
    while (i + 1 < locfiles.Count);
    rtb_info.Text = "文件目录遍历成功! ";
    return true;
}
```

使用循环方式完成树型结构的目录及文件信息的获取，在取得列表后，接下来就是此过程的一个重要步骤：

(2) 每次只处理单个文件的上传：

如果是目录，则判断 FTP 上目录是否存在，如果不存在，只要建立相应的目录即可：

```
    public Boolean fun_upfile(string as_file, Boolean ab_Dir,
string as_locpath ) {
        string ls_ftpfile;
        Boolean lb_find;
        ls_ftpfile = as_file;
        ls_ftpfile = ls_ftpfile.Replace(as_locpath, is_ftpRoot);
        ls_ftpfile = ls_ftpfile.Replace("\\", "/");
        if (ab_Dir){ //目录需要建立
            FtpDll.FtpCreateDirectory (hInternetConnection,
ls_ftpfile);
            return true;
        }
        lb_find = FtpDll.FtpPutFile (hInternetConnection,
as_file, ls_ftpfile, FTP_TRANSFER_TYPE_BINARY, 0);
        return lb_find;
        }
```

可以看出对于单文件上传其实非常简单，只要使用 dll 中对应的函数 FtpPutFile 就可以完成了。

对于文件的上传和下载在功能上都已经完成了，相对来说，再来实现比对功能就不是什么复杂的问题了，只是过程会更复杂一些。

2.2.3 文件比对

文件比对其实整合了文件下载和本地文件读取的功能，它要比较相应的文件在 FTP 是否正确上传，所以检测 FTP 上的文件是否和本地相同，而本地删除的文件并不处理。命令使用方法如下：

Myftp -c d:\his3 his3/his3@192.168.1.100:21/his3

实际使用效果如图 3 所示。

在实现时，连接 FTP 后取 FTP 上对应目录，所以循环也

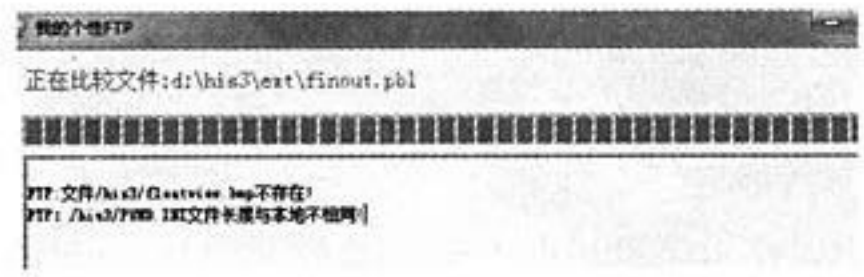


图 3 文件比对

以本地为主。由于本地与 FTP 上有差异的可能有多种情况，首先已经取得 FTP 对应目录下的文件与目录信息，所以：

(1) 文件或目录在 Ftp 上不存在，可以利用 ftpdll.fun_find ftp 函数在 ftpfiles 中直接找出。

(2) 如果发现目录不存在，自动从 locfiles 中剔除了同一目录及子目录下的所有文件及目录（不需要再作比较），只需要告知目录不存在即可。

(3) 在文件存在时，打开 FTP 文件句柄后取得长度后本地文件进行比较，发现长度不同，则文件不同。

(4) 如果长度相同，此时就可以根据传入参数来决定是整个文件与本地文件进行比较还是读取限定长度来进行比较。

在函数 fun_FtpCompare 中完成本地目录和 FTP 目录之间的文件比较，是否正确上传：

```
public Boolean fun_ftpCompare (string strLoc, string strFtp,
Int32 numSize ){
        int     li_which, li_countfiles = 1;
        int     li;
        int     li_len;
        string  ls_path;
        Boolean lb_test;
```

```
FtpDll MyFtp = new FtpDll();
if (! fun_locfiles(strLoc))
   return false;
MyFtp.fun_Connect(strFtp);
li_countfiles = MyFtp.fun_FtpDirs();
li_countfiles = locfiles.Count;
pb_info.Maximum = li_countfiles;
pb_info.Step = 1;
 for (li_which = 0; li_which < li_countfiles;
li_which++){
            rtb_info.Text = " 正在比较文件:" + locfiles
[li_which].cFile;
            pb_info.PerformStep();
            Application.DoEvents();
            //有文件才进行处理
            if (locfiles[li_which].cFile.Length == 0) continue;
            lb_test = MyFtp.fun_Compare (locfiles[li_which].
cFile, locfiles[li_which].bDir, strLoc, numSize, rtb_result);
            //如果目录不存在,则此子目录下的所有文件就直
//接跳过
            if ( locfiles[li_which].bDir & ! lb_test ){
               ls_path = locfiles[li_which].cFile;
               li_len = locfiles[li_which].cFile.Length;
               for ( li = li_which + 1; li < li_countfiles; li ++ ){
                  if (locfiles[li].cFile.Length < li_len)
                     continue;
                  if ( locfiles [li].cFile.Substring (0, li_len) ==
locfiles[li_which].cFile )
                     locfiles[li].cFile = "";
               }
            }
         }
         return true;
      }
```

可见结合文件下载和上传，使用 MyFtp.fun_FtpDirs 取得了 FTP 上的文件列表，然后依次比较文件夹和文件是否存在，及相应的文件是否相同。

由于考虑到有的文件实在过于巨大，因此在检测时，还提供了抽样进行比较的方法，就是可以读取一个大文件的指定长度来进行快速检测。它的命令行是：

Myftp –c –q128000 d:\his3 his3/his3@192.168.1.100:21/his3

表示对文件最多只抽取 128000 字节进行检测，这样当大文件很多时，就不需要从 FTP 上读取文件的全部内容来进行比较，完成文件的快速比较，虽然快速比较不一定完全正确，但是有时可以进行临时性的快速比较，节约比较时间，相当于快速测试一样。

因此在文件比较函数 fun_Compare 中，提供了全文件的比较和抽取部分长度的比较方法：

```
//2010-01-25 比对相应的文件时用到的 4 个参数:本地
//文件,是否目录,本地路径,提示内容
      public Boolean fun_Compare (string as_file, Boolean
ab_dir, string as_locpath, Int32 numSize, System.Windows.
Forms.RichTextBox atextbox ){
         Int32  nBytesRead = 0;
         Int32  nLeft, i;
         Int32  MAX_BYTES_TO_READ = 1024;
         UInt32 ll_filesize, ll_fileother; //文件长度
         Int32  ll_needread;
            byte[] lb_buffer = new byte[MAX_BYTES_
TO_READ];
            byte[]  lb_fileread = new byte[MAX_BYTES_
TO_READ];
         IntPtr  lp_Handle;
         Boolean lb_read;
         Boolean lb_same = true;
         string  ls_ftpfile;
         string  ls_info;
         itextbox = atextbox;
         ls_ftpfile = as_file;
         ls_ftpfile = ls_ftpfile.Replace(as_locpath, is_ftpRoot);
         ls_ftpfile = ls_ftpfile.Replace("\\", "/");
         if ( ! fun_findinftp( ls_ftpfile ) ){
            ls_info = "FTP:文件";
            if ( ab_dir )
               ls_info = "FTP:目录>>>";
            fun_dispinfo( ls_info + ls_ftpfile +"不存在! " );
            return false;
         }
         if (ab_dir)
            return true;
                  lp_Handle = FtpDll.FtpOpenFile(hInternet
Connection, ls_ftpfile, GENERIC_READ, FTP_TRANSFER_TY
PE_BINARY, 0);
         if (lp_Handle == IntPtr.Zero){
            fun_dispinfo (" 打开 FTP 文件:" + ls_ftpfile + "失
败! ");
            return false;
         }
         ll_fileother = 0;
          ll_filesize = FtpDll.FtpGetFileSize (lp_Handle, ref
ll_fileother);
         if (ll_filesize <= 0){
            FtpDll.InternetCloseHandle(lp_Handle);
            return false;
         }
          FileStream fsFile = File.Open (as_file, FileMode.
Open, FileAccess.Read);
         if ( fsFile.Length ! = ll_filesize ){
            fun_dispinfo( "FTP:"+ls_ftpfile+"文件长度与本地
不相同! ");
```

```
        lb_same = false;
        goto DoEnd;
    }
    nLeft = numSize;
    do{
        ll_needread = MAX_BYTES_TO_READ;
        if (nLeft < ll_needread && numSize >0 )
            ll_needread = nLeft;
        lb_read = InternetReadFile  (lp_Handle, lb_buffer,
MAX_BYTES_TO_READ, ref nBytesRead);
            if (ll_needread < nBytesRead)
                ll_needread = nBytesRead;
            fsFile.Read(lb_fileread, 0, ll_needread);
            for (i = 0; i < ll_needread; i++){
                if (lb_buffer[i] ! = lb_fileread[i]){
                    lb_same = false;
                    break;
                }
            }
            nLeft = nLeft – nBytesRead;
            if (nLeft <= 0 && numSize>0 )
                nBytesRead = 0;
        }
        while (nBytesRead > 0);
            if (! lb_same)
                fun_dispinfo("FTP 文件：" + as_file + "与本地文件
内容有差异！");
        DoEnd:
            fsFile.Close();
            FtpDll.InternetCloseHandle(lp_Handle);
            return lb_same;
        }
```

所以在文件比较对时，长度不同或者是文件不存在时，可以即时返回，而对于长度相同时就需要读取内容进行比较，有限定长度时，只比对指定长度（大于文件长度时，只取文件长度）。

3 补充说明

介绍了使用 Windows 系统级 DLL 对 FTP 调用的功能，它不仅可以完成用户对 FTP 更新和下载功能的使用，而且还可以进行 FTP 的比对和快速比对，使 FTP 的维护变得更为简单易用。

当然在实现了上下传和比对功能后，就可以在此基础上，打造更多实用功能，去处理其他更复杂的问题了。

由于此功能具有极强的实用性，因此在进行 FTP 维护时就具有很好的应用价值。

（收稿日期：2012-08-01）

（上接第 53 页）

```
else if(stremp1[0]=='7')
{stremp2 =stremp1;stremp2.Delete (0);m [7] =1;score [7] =atoi
(stremp2);if  (stremp2  [0]  =='0')  {stremp2.Delete  (0);}
SetDlgItemText(IDC_EDIT8,stremp2);}
else if(stremp1[0]=='8')
{stremp2 =stremp1;stremp2.Delete (0);m [8] =1;score [8] =atoi
(stremp2);if  (stremp2  [0]  =='0')  {stremp2.Delete  (0);}
SetDlgItemText(IDC_EDIT9,stremp2);}
else if(stremp1[0]=='9')
{stremp2 =stremp1;stremp2.Delete (0);m [9] =1;score [9] =atoi
(stremp2);if  (stremp2  [0]  =='0')  {stremp2.Delete  (0);}
SetDlgItemText(IDC_EDIT10,stremp2);}
}
//以上是把接收到的数据进行归类并在各个不同的编辑框中显
//示
x=0;
for(i=0;i<10;i++)
{
    if(m[i]){x++;s+=score[i];if(max<score[i])max=score[i];if
(min>score[i])min=score[i];}
}
//找出最大、最小值
if(x>=sf)
//已获得全部分数
{
Float sss;
    sss=(float)(s-max-min)/(x-2);
//求出去掉最大、最小值后的平均值
  stremp2.Format("%d",sss);
  SetDlgItemText(IDC_EDIT0,stremp2);
    stremp2.Format("%d",max);
    SetDlgItemText(IDC_EDIT11,stremp2);
  stremp2.Format("%d",min);
    SetDlgItemText(IDC_EDIT12,stremp2);
//将最大、最小和平均值进行显示
}
```

5 结语

这是一个课外制作，主要以 C 和 C++结合电路知识进行的一次动手实践，其中，遇到了许多书本上没有提到的困难，在一一排除这些难题后，实现了 50 米远距离无线打分通信。最后与大家分享制作感受：在编写程序时主要的是理清整体思路，不能弄到哪算哪。要模块化编程，这样在遇到复杂问题的时候才更容易解决，在对待一个复杂问题时更要步步为营，注意细节，细节上不注意就很容易失败。往往一个复杂问题的解决方法并没有想象的复杂，从最基础的做起才是解决问题的关键。

（收稿日期：2012-07-16）

PHP 多线程抓取多个网页及获取数据的通用方法

刘洪志

摘　要：从允许外链的网络相册中得到图片的外链地址，然后用于自己的博客、网站，是网站管理者经常碰到的问题，大部分网络相册提供便捷操作以满足用户需求，但是有些网络相册没有提供便捷操作。从实例出发，探讨使用 PHP 中 CURL 的多线程获取网路相册的图片外链地址通用方法。

关键词：PHP 环境；多线程；CURL 多线程；正则表达式；网络相册

1　问题提出

PHP 环境下抓取多个网页可以参考使用 fopen ()、file_get_contents () 等函数加以实现，但这种方式通常是顺序处理，即利用循环依次读入多个网页，然后再分析处理数据，当网页个数较少时这不失为一种简便有效的方式，但是当要处理的网页个数很多时，就会带来致命性的问题，因为 PHP 环境下执行 PHP 代码都有时间限制。此时多线程获取多个网页就成为解决此类问题的不二之选。

2　处理此类问题的一般过程

（1）把要多线程处理的多个网页的 URL 放入数组。

（2）使用 CURL 的多线程处理函数读取多个网页数据。

（3）使用正则表达式从获取到的多个网页数据中提取有用数据。

3　实例

3.1　实际问题

Adobe 公司提供了 2GB 空间的免费网络相册，可喜的是该相册允许用户外链，以 Adobe 公司的实力，相信它可以很稳定地提供此类服务，用户得到某张图片外链地址的方法也很简单，但是要想一次获取多张图片的外链地址却近乎于一个不可能完成的任务。

3.2　解决方法

经简单分析该网络相册的 HTML 代码后发现：HTML 代码中包含相册图片的外链地址，只需要使用正则表达式从 HTML 代码中提取出图片的外链地址即可。

但下一个问题随之出现了：该相册的每个页面中仅显示 18 张图片，如果一个相册中有几百张图片，那么需要抓取的网页至少是几十个，为了提高效率，需要使用 CURL 多线程抓取解决之。

3.3　PHP 源代码及解释

```
<html>
 <head>
  <title>获取 PhotoShop 相册的图片外链 URL – 任丘市职教中心 – 刘洪志 – 2012/05/15</title>
  <meta http –equiv = "Content –Type" content = "text/html; charset=gb2312" />
 </head>
 <body>
<?php
  // 检查用户是否提交了数据
  if ($_SERVER['REQUEST_METHOD'] ! = 'POST')
  {
   $me = $_SERVER['PHP_SELF'];
/*******************************
```

用户尚未提交数据，则构造一个表单，要求用户提供已共享相册的 URL 和该相册的总页数。

已共享相册的 URL 从浏览器的地址栏中得到，例子：

http://www.photoshop.com/users/LiuHongZhi/albums/80874394f7604cc7abebbe40d80d9b18

相册的总页数在页面的下部，例子：

在浏览器打开上面地址对应的相册，可以在页面下部看到 1 of 27，27 就是该相册的总页数

```
*******************************/
?>
  <form name = "ps_url" method = "post" action = " < ?php echo $me;?>">
   <table border="0" cellspacing="0" cellpadding="2">
    <tr>
      <td><h2>获取 PhotoShop 相册的图片外链 URL –职教中心 – 刘洪志 – 2012/05/15</h2></td>
    <tr>
     <td > 相 册 URL: <input type = "text" size = "120" name = "url" value = "" onMouseOver =this.focus () onFocus = this.select()></td>
```

```
        </tr>
        <tr>
          <td>相册页数：<input type="text" size="4" name="
page" value="" onMouseOver=this.focus() onFocus=this.
select()></td>
        </tr>
        <tr>
          <td><input type="Submit" name="Submit" value="提
交"></td>
        </tr>
      </table>
    </form>
<?php
  }
  else //// 用户已提交数据
  {
    //// 此处可以加时间测试代码，记录开始时间
    //// 构造访问相册中各个页面的 URL，并放入数组中
    for ($p=1; $p<=$_POST['page']; $p++)
    {
      $urls[] = $_POST['url'] . "?page=" . $p;
    }
    //// 开始多线程获取网页数据
    //// 创建批处理 CURL 句柄
    $mh = curl_multi_init();
    //// 设置 CURL 传输选项
    foreach ($urls as $i => $url)
    {
      $conn[$i]=curl_init($url);
      //// 获取的信息以文件流的形式返回
curl_setopt($conn[$i],CURLOPT_RETURNTRANSFER,1);
      //// 向 CURL 批处理会话中添加单独的 CURL 句柄
      curl_multi_add_handle ($mh,$conn[$i]);
    }
    //// 执行批处理句柄
    $active = null;
    do
    {
      $mrc = curl_multi_exec($mh,$active);
    }while ($mrc == CURLM_CALL_MULTI_PERFORM);
    while ($active and $mrc == CURLM_OK)
    {
      if (curl_multi_select($mh) != -1)
      {
        do
        {
          $mrc = curl_multi_exec($mh, $active);
        }while ($mrc == CURLM_CALL_MULTI_PERFORM);
      }
    }
    //// 把获取的多个网页信息合并放到 $contents 变量中
    $contents = "";
    foreach ($urls as $i => $url)
    {
  $contents = $contents . curl_multi_getcontent($conn[$i]);
      // 关闭 CURL 资源，并且释放系统资源
      curl_close($conn[$i]);
    }
    //// 多线程获取网页数据结束
    //// 此处可以加时间测试代码，记录结束时间
/*************************************
    利用正则表达式从获取的网页代码中提取图片外链 URL
相册的原始 HTML 部分代码如下：
<img alt="" onerror="fixThumb(this, 'm')" src="http://api.
photoshop.com/v1.0/accounts/
b5831c03ea9940fe9e15f358ee894dbb/assets/
85060af17f8c40bb9c0f37eb75258d9a/renditions/256.jpg" />
    上述 HTML 代码中包含图片的外链 URL：
http://api.photoshop.com/v1.0/accounts/
b5831c03ea9940fe9e15f358ee894dbb/assets/
85060af17f8c40bb9c0f37eb75258d9a
    每个单独页面的 HTML 代码中最多有 18 处此类代码，因
此需要使用 preg_match_all() 函数和正则表达式获取有用数
据，并放入数组中
*************************************/
    preg_match_all('/\'m\')" src="(http:\/\/api\.photoshop\.
com\/v1\.0\/accounts\/[a-fA-f0-9]{32}\/assets\/[a-fA-f0-9]
{32}\/renditions\/256\.jpg"/',$contents, $result, PREG_SET_
ORDER); // 取出图像的 URL
    //// 把图片的外链地址输出到浏览器
    for($j=0; $j<count($result); $j++)
    {
      //// 你可以根据需要更改输出格式
      echo $j+1 . "、" . $result[$j][1] . "<br>";
    }
  }
?>
 </body>
</html>
```

4 测试多线程获取网页的效率

4.1 服务器环境

至强 1.8GHz，内存 1GB，apache + PHP，100MB 网通光纤。

4.2 测试方式

使用两台配置相同、网路环境相同的计算机，同时提交数据，分别测试多线程获取及使用 file_get_contents() 函数顺序获取所耗费的时间。

下面给出使用 file_get_contents() 函数顺序获取多网页数据的代码：

```
<html>
```

```
 <head>
  <title>获取 PhotoShop 相册的图片外链 URL – 任丘市职教中心 – 刘洪志 – 2012/05/15</title>
  <meta http-equiv = "Content-Type" content = "text/html; charset=gb2312" />
 </head>
 <body>
<?php
  if ($_SERVER['REQUEST_METHOD'] ! = 'POST')
  {
   $me = $_SERVER['PHP_SELF'];
?>
   <form name = "ps_url" method = "post" action = " < ?php echo $me;?>">
    <table border="0" cellspacing="0" cellpadding="2">
     <tr>
          <td><h2>获取 PhotoShop 相册的图片外链 URL – 任丘市职教中心 – 刘洪志 – 2012/05/15</h2></td>
     <tr>
      <td > 相 册 URL: <input type = "text" size = "120" name = "url" value = "" onMouseOver =this.focus () onFocus = this.select()></td>
     </tr>
     <tr>
      <td>相册页数:<input type="text" size="4" name="page" value = "" onMouseOver =this.focus () onFocus =this.select()></td>
     </tr>
     <tr>
<td><input type="Submit" name="Submit" value="提交"></td>
     </tr>
    </table>
   </form>
<?php
 }
 else
 {
  $contents ="";
  //// 此处可以加时间测试代码,记录开始时间
  for ($i=1; $i<=$_POST['page']; $i++)
  {
   $ps_url=$_POST['url'] . "?page=" . $i;
   $contents =$contents . file_get_contents($ps_url);
      }
  //// 此处可以加时间测试代码,记录结束时间
  preg_match_all('/\'m\"\)" src="(http:\/\/api\.photoshop\.com\/v1\.0\/accounts\/[a-fA-f0-9]{32}\/assets\/[a-fA-f0-9]{32}\/renditions\/256\.jpg"/',$contents, $result, PREG_SET_ORDER);
// 取出图像的 URL
  for($j=0; $j<count($result); $j++)
  {
   echo $j+1 . "、" . $result[$j][1] . "<br>";
  }
 }
?>
 </body>
</html>
```

4.3 测试执行时间代码

在多线程获取网页数据前加入以下代码：

```
$stime=microtime(true); //获取程序开始执行的时间
```

在利用正则表达式从获取的网页代码中提取图片外链URL之前加入以下代码：

```
$etime=microtime(true); //获取程序执行结束的时间
$total=$etime-$stime; //计算时间差
echo " 起始时间:{$stime}, 终止时间:{$etime}, 执行时间:{$total} 秒<br>";
```

利用这4行代码可以简单测算代码执行时间。

4.4 测试对象

相 册 URL:http://www.photoshop.com/users/LiuHongZhi/albums/80874394f7604cc7abebbe40d80d9b18

总页数:27

4.5 测试结果

共测试5组，数据如下：

多线程方式

起始时间:1337214234.0938,终止时间:1337214239.6254,执行时间:5.5316519737244 秒

起始时间:1337214360.9688,终止时间:1337214365.3676,执行时间:4.3988430500031 秒

起始时间:1337214454.7813,终止时间:1337214458.9861,执行时间:4.2048180103302 秒

起始时间:1337214585.6094,终止时间:1337214591.7311,执行时间:6.1217668056488 秒

起始时间:1337214783.5,终止时间:1337214789.0528,执行时间:5.5528419017792 秒

顺序方式

起始时间:1337214234.1406,终止时间:1337214300.4417,执行时间:66.301088809967 秒

起始时间:1337214360.9844,终止时间:1337214422.0441,执行时间:61.059705018997 秒

起始时间:1337214454.8125,终止时间:1337214534.0571,执行时间:79.244569063187 秒

起始时间:1337214585.5625,终止时间:1337214651.4874,执行时间:65.924920082092 秒

起始时间:1337214783.1719,终止时间:1337214878.4579,执行时间:95.286026000977 秒

4.6 结果分析

多线程获取网页的时间取决于某一个最慢网页，与网页数量无关，而顺序获取网页则是全部网页的时间总和，这也就是测试结果中两种方式的巨大差异的根源。

(下转第64页)

编程实现在曲线上配置小短线

穆宣社

摘　要：堑壕图标的绘制关键是在定位线上配置两个垂直小线段。分析了小短线计算的思路，介绍了 VC++计算机标图软件在曲线上配置小短线的计算与绘制方法。

关键词：计算机标图；短线；配置

计算机标号的显示难点是描述它的变化性，矢量标号是变化功能最完备，也是实现最复杂的图标，它从形状上可划分为点、线、面 3 种类型，从变化性质上可分为固定、随机两种类型，从实现手段上可分为数据标号和函数标号。矢量堑壕图标显示的实现属于线状随机函数标号，因其复杂性，特此介绍实现方法。

1　显示带小短线的曲线思路

绘制该标号时，需要给出一系列定位点列坐标，该符号以这些点列作为定位线绘制，并按照等间距 ΔS 在定位线上配置两个垂直小线段，假设这些定位点都是通过光滑处理程序处理过的间距很小的点列。该符号绘制的关键问题是配置两垂直的小短线，这些小的短线在光滑曲线上的距离是相等的，如图 1 中各个短线沿光滑曲线的距离是 160，是用量测工具测量出来的距离。即求出如图 1 上 D 点的坐标，求解公式如下：

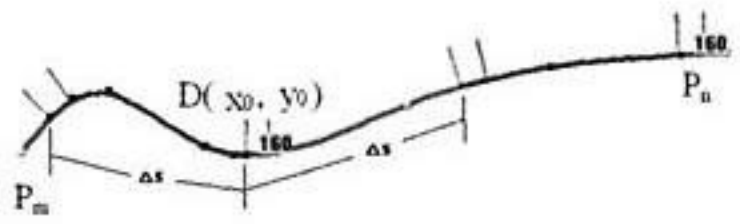


图 1　曲线上短线的绘制示例

$$S=\sum_{i=1}^{m}\sqrt{(X_{i+1}-X_i)^2+(y_{i+1}-y_i)^2}$$

当 S 刚刚大于 ΔS 时，D 点在 Pm 与 Pm+1 之间

这时：

$$X_D=\frac{(S-\Delta S)}{\sqrt{(X_{m+1}-X_m)^2+(y_{m+1}-y_m)^2}}(X_{m+1}-X_m)$$

$$Y_D=\frac{(S-\Delta S)}{\sqrt{(X_{m+1}-X_m)^2+(y_{m+1}-y_m)^2}}(y_{m+1}-y_m)$$

2　采集曲线控制点坐标

判断鼠标单击双击的标准：（prePoint.x == point.x && prePoint.y == point.y)

如果是单击（上述判断为 false），进行数据采集：

```
CArray <CPoint, CPoint> Mcoor;
CSpaintView::OnLButtonUp 在单击时放入数组,pDoc->m_pCurMark->Mcoor.Add(point);
HGLOBAL hCodi = GlobalAlloc (GMEM_MOVEABLE | GMEM_ZEROINIT,(DWORD)(pNum + 3)*sizeof(POINT));
if (hCodi)  {
 POINT* pCodi = (POINT*)GlobalLock(hCodi);
 for (i = 0; i < pNum; i++){
      CPoint SP  (pDoc ->m_pCurMark ->Mcoor [i].x, pDoc->m_pCurMark->Mcoor[i].y);
      SP -= pDoc->Org;  *(pCodi + i) = SP;}
}
```

如果是双击（上述判断为 true），结束数据采集，进行数据处理，绘制图形 DrawMark ()。

3　计算曲线上短线的坐标

这里用四边形形表示短线，调用 computecood（）计算四边形的坐标。

```
/*****************************************************/
/*  函数的参数说明
/*  mbuff :输入数据                  */
/*  n     :输出数据个个数这里=4     */
/*  buff1  :参数控制                 */
/*  buff2  :输出数据根据 n 确定,这里用四边形绘制,输出的点的个数=4 */
/*****************************************************/
void computecood (double * mbuff, int n, short* buff1, LPINT buff2)
{
int     trax,tray,ii;
double      xx,yy;
trax = (int)*(mbuff+4);
tray = (int)*(mbuff+5);
switch((int)mbuff[6])
{case 0:neg1 = 1;      break;
 case 1:neg1 = mbuff[0] >= 0. ? 1 : -1;break;
 case 2:neg1 = mbuff[0] >= 0. ? -1 : 1;break;
```

```
 case 3:neg1 = -1;break;
}
if (*(mbuff) == 1.)
 {
    if (*(mbuff+2) == 1. && *(mbuff+3) == 1.)
    {
            for (ii = 1 ; ii < n+n ; ii += 2)
            {
                *(buff2+ii-1) = *(buff1+ii-1) + trax;
        *(buff2+ii) = tray - (int)(*(buff1+ii) * neg1);
            }
        }
        else
        {
            for (ii = 1 ; ii < n+n ; ii += 2)
            {
*(buff2+ii-1) = (int)(*(buff1+ii-1) * *(mbuff+2)) + trax;
*(buff2+ii) = tray - (int)(*(buff1+ii) * *(mbuff+3) * neg1);
                int r1,y1;
                r1=*(buff2+ii-1) ;
                y1=*(buff2+ii);
            }
        }
   }
   else
   {
        if (*(mbuff+2) == 1. && *(mbuff+3) == 1.)
        {
            for (ii = 1 ; ii < n+n ; ii += 2)
            {
                xx = *(buff1+ii-1);
                yy = *(buff1+ii) * neg1;
*(buff2+ii-1) = (int)(xx * *mbuff - yy * *(mbuff+1)) + trax;
*(buff2+ii)   = tray - (int)(xx * *(mbuff+1) + yy * *mbuff);
            }
        }
        else
        {
            for (ii = 1 ; ii < n+n ; ii += 2)
            {
                xx = *(buff1+ii-1) * *(mbuff+2);
                yy = *(buff1+ii) * *(mbuff+3) * neg1;
   *(buff2+ii-1) = (int)(xx * *mbuff - yy * *(mbuff+1)) + trax;
*(buff2+ii)   = tray - (int)(xx * *(mbuff+1) + yy * *mbuff);
                }
        }
    }
    mbuff[6] = (float)4;
  }
```

4 绘制多边形（短线）

（1）绘制主要基线，不包括垂直的短线。

::SelectObject (hDC, OCsig ? (hPenT = ::ExtCreatePen (roundstyle,ownthick,&LogBrush,0,NULL)) : hPen2);

::Polyline(hDC, (LPPOINT)lppbf, n);

（2）绘制短线。用 4 个点组成一个多边形来绘制这个短线，这样做的好处是可以控制该短线的宽度，如图 2 所示的黑色①②③④部分。利用 4 个点来绘制短线的长度和宽度。绘制结果如图 2 下所示。其中绘制函数的参数 N=4。

```
::SelectObject(hDC, OCsig ? hPen0A[lpb2[i]&0xf] : hPen0);
::SelectObject(hDC, (OCsig ? hBrushA[lpb2[i]&0xf] : hBrush1));
::Polygon(hDC, (LPPOINT)lppbf, n);//绘制多边形
```

间距与长短不同的小短线显示。宽度逐渐增宽，图 2 中小短线长度逐渐加大。

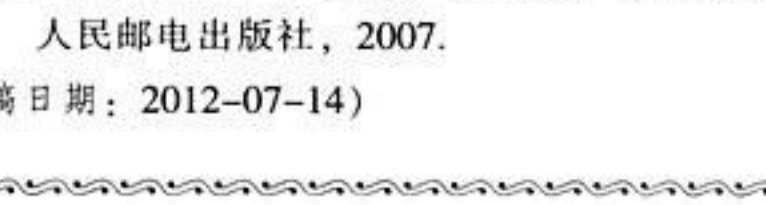


图 2

5 结语

电脑游戏的态势显示中，作战行动完全依靠军标来表达，因此军标显示是一项非常重要的功能。军标显示技术包括了二维军标显示和三维军标显示，其中二维军标又包括了几种不同类型的显示军标，文中介绍的带有短线的曲线不仅适用于堑壕绘制，对于警戒壕沟、铁丝网等工程类标号也有借鉴意义。

参考文献

[1] ShirleyP. 计算机图形学 [M]. 高春晓，赵清杰，等，译. 北京：人民邮电出版社，2007.

（收稿日期：2012-07-14）

（上接第 62 页）

5 结语

通过以上实例代码，看到 PHP 环境中使用 CURL 多线程获取多网页的效率比顺序获取多网页的效率高出很多，这在对 PHP 执行时间有严格要求的服务器中是必不可少的技术手段。这段实例代码体现了处理此类问题的一般过程及步骤，读者可以根据网站的不同及获取内容的不同，稍加改动即可得到满意结果。

参考文献

[1] PHP 用户手册. http://cn2.php.net/manual/zh/book.curl.php.

（收稿日期：2012-09-05）

基于 OpenGL 实现 NURBS 汽车前盖曲面造型

蔡智明

摘　要：分析了 NURBS 构造自由曲面的方法，结合汽车前盖曲面流线型知识，基于 OpenGL 技术，利用 VC++开发工具进行汽车前盖曲面的造型。

关键词：NURBS 方法；汽车曲面造型；OpenGL 库；VC++语言

1　NURBS 简介

曲面造型技术是计算机辅助设计 CAD 和计算机图形学 CG 中最为关键的学科分支之一，1975 年由 Versprille 在他的博士论文中首先提出了非均匀有理 B 样条。NURBS 这个概念随着对曲线曲面实体造型技术的不断深入研究，非有理与有理的 Bezier 曲线曲面形式和非有理的 B 样条曲线曲面形式都被统一到 NURBS 形式之中，1991 年国际标准化组织 ISO 颁布的关于工业产品数据交换的 STEP 国际标准，把 NURBS 定义为工业产品几何形状的唯一数学方法。目前可以说 NURBS 方法已经成为了曲面造型技术中最重要的方法。

2　NURBS 的曲线和曲面方程

K 次 NURBS 曲线的数学表达式：

$$C(t)=\frac{\sum_{i=0}^{n}P_i\omega_i N_{i,k}(t)}{\sum_{i=0}^{n}\omega_i N_{i,k}(t)}=\sum_{i}^{n}P_i R_{i,k}(t)$$

其中，$\{P_i\}$ 为控制点；$\{\omega_i\}$ 为控制点的权因子；$N_{i,k}(t)$为 k 阶 B 样条基函数。

U 向 p 阶，v 向 q 阶的 NURBS 曲面的数学表达式：

$$S_{(u,v)}=\frac{\sum_{i=0}^{n}\sum_{j=0}^{m}P_{i,j}\omega_{i,j}N_{i,p}(u)N_{j,q}(v)}{\sum_{i=0}^{n}\sum_{j=0}^{m}\omega_{i,j}N_{i,p}(u)N_{j,q}(v)}=\sum_{i=0}^{n}\sum_{j=0}^{n}R_{i,j}(u,v)P_{i,j}$$

其中，$\{P_i\}$ 为曲面网络控制点；$\{\omega_i\}$ 为曲面网络控制点的权因子；$N_{i,p}(u)$，$N_{j,q}(v)$分别为 p 阶和 q 阶的 B 样条基函数。

3　NURBS 方法的特点

NURBS 方法的提出，一个主要目的是为了找到既能与描述自由型曲线曲面的 B 样条方法相统一，又能精确表示二次曲线弧和二次曲面的方法。相比于非有理贝塞尔方法和非有理 B 样条方法具有如下优点：

(1) 对规则曲线曲面（如圆锥曲线、二次曲线和曲面等）和自由型曲线曲面提供了统一的数学表示，便于用一个统一的数据库来存取这两类形状信息。

(2) 具有更多的形状控制自由度。由于选择的节点值可以不同，节点值间的距离也可以不一样，非均匀 B 样条曲线在曲线形状的控制方面就更加方便，通过节点向量的不同间距，就可以在不同的区间上得到不同的基函数，用以调整曲线的形状。另一方面，NURBS 曲线还可以通过控制点和权因子来灵活地改变曲面的形状。

(3) NURBS 曲线在平移、比例、旋转和透视投影变换下是不变的，即曲线的控制顶点经过平移、比例、旋转和透视投影变换后生成曲线（或曲面），与在生成曲线（或曲面）后进行上述变换，二者是等价的。

(4) 可以用少量的控制点就可以构造出整个曲面。当然，NURBS 方法也有其不足之处，主要在于它比一般的曲线、曲面定义方法更费存储空间和处理时间，权因子选择不当会造成曲线曲面形状的畸变等。

4　程序实现

4.1　使用 OpenGL 的准备

OpenGL 是一种过程性而不是描述性的图形 API，可以在计算机上调用 OpenGL 绘制函数，就能够实现二维或者三维图形的绘制，即便用某种编程语言可以是 VC，也可以是别的语言如 VB、Delphi 编写的还须要调用一个或者多个 OpenGL 库。

基于 OpenGL 标准开发的应用程序必须运行于 32 位的 Windows 平台下而且运行时还须要有动态链接库 OpenGL32.DLL、Glu32.DLL 这两个动态链接库可以在 Windows 的 system32 的目录下找到，如果是低版本的 Windows 用户可能须要手工加载，这里使用的是 VC 来进行开发，需要在环境设置中的 Link 选项组 Lib 列表框加入 opengl32.lib,glu32.lib，若需使用 OpenGL 的辅助函数还要加上 glaux.lib。

基于 OpenGL 的 Windows 应用程序中的初始头文件要包括：

```
#include <gl/gl.h>
```

```
#include <gl/glu.h>
#include <gl/glut.h>
```

如果须要用到辅助库函数还要加上 # include <gl/glaux.h>

4.2 使用 OpenGL 实现 NURBS 曲面

OpenGL 使用一种称为求值技术 evaluator 来进行曲线曲面的绘制并且可以自动计算法向量在绘制 NURBS 曲面方面 OpenGL 的 GLU 提供了一个 NURBS 接口，有如下函数：

GLUnurbsObj* glNewNurbsRenderer（）创建一个 NURBS 对象，并返回一个指向该对象的指针。如果没有足够的内存分配给该对象，则返回值为 0。

void gluNurbsProperty (GLUnurbsObj* nobj, GLenum property, GLfloat value) 设置 NURBS 属性。

nobj 指向 NURBS 对象的指针，property 需设置的属性，value 设置指定属性的值。

glBeginSurface 及 gluEndSurface 两个函数一起限定一个 NURBS 面的定义。返回值均为 void，参数均为 GLUnurbsObj* nobj，为指向 NURBS 对象的指针：

void gluNurbsSurface (GLUnurbsObj *nobj, Glint knot_count, GLfloat tknot_count, GLfloat *tknot, Glint s_stride, Glint t_stride, GLfloat *ctlarry, GLint sorder, GLint torder,GLenum type)

定义 NURBS 曲面形状。

nobj 指向 NURBS 对象的指针，sknot_count 参数化 u 方向上的节点数，sknot 参数化 u 方向上的非递减节点值，tknot_count 参数化 v 方向上的节点数，tknot 参数化 v 方向上的非递减节点值，s_stride 在 ctlarry 中参数化 u 方向上相邻控制点的偏移量，t_stride 在 ctlarry 中参数化 v 方向上相邻控制点的偏移量，ctlarryNURBS 的控制点数组，sorder 参数化 u 方向上 NURBS 的阶数，阶数比维数大 1，torder 参数化 v 方向上 NURBS 的阶数，阶数比维数大 1，type 曲面类型。

glutSwapBuffers 函数交换了前后缓冲区，函数原型如下：

void glutSwapBuffers ()。

4.3 绘制曲面

编写程序的基本步骤如下：

（1）利用 VC++创建 Win32 程序，并导入需要的 OpenGL 头文件。

（2）初始化 OpenGL 的绘图环境，包括定义颜色格式和缓冲模式、设置光源、设置材质、定义投影方式.主要代码如下：

```
void myInit(void)
{
    glClearColor(1.0,1.0,1.0,0.0);//设置背景色
    /* 为光照模型指定材质参数 */
    glMaterialfv(GL_FRONT,GL_DIFFUSE,mat_diffuse);
    glMaterialfv(GL_FRONT,GL_SPECULAR,mat_specular);
    glMaterialfv(GL_FRONT,GL_SHININESS,mat_shininess);
    glLightfv(GL_FRONT,GL_POSITION,light_position);//设置
//光源参数
        glLightModeli (GL_LIGHT_MODEL_TWO_SIDE,
GL_TRUE);//设置光照模型参数
    /* 激活光照 */
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glDepthFunc(GL_LEQUAL);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_LEQUAL);
    glEnable(GL_AUTO_NORMAL);
    glEnable(GL_NORMALIZE);
    /* 设置特殊效果 */
                    glBlendFunc (GL_SRC_ALPHA,
GL_ONE_MINUS_SRC_ALPHA);
    glHint(GL_LINE_SMOOTH_HINT,GL_DONT_CARE);
    glEnable(GL_BLEND);
    glFrontFace(GL_CW);
    glShadeModel(GL_SMOOTH);
    glEnable(GL_LINE_SMOOTH);
    theNurb = gluNewNurbsRenderer();//创建 NURBS 对象
//theNurb
                    gluNurbsProperty (theNurb,
GLU_SAMPLING_TOLERANCE,25.0);
    gluNurbsProperty        (theNurb,GLU_DISPLAY_MODE,
GLU_FILL);//实体填充式
}
```

（3）利用 OpenGL 中的 NURBS 函数绘制和显示曲面，主要代码如下：

```
void myDisplay(void)
{
                        GLfloat knots [10] =
{0.0,0.0,0.0,0.0,0.0,1.0,1.0,1.0,1.0,1.0};
                    glClear(GL_COLOR_BUFFER_BIT|
GL_DEPTH_BUFFER_BIT);
    glRotatef(50.0,1.0,1.0,0.0);
    /* 曲面 */
    glPushMatrix();
    glTranslatef(0.0,0.0,0.0);
    gluBeginSurface(theNurb);//开始绘制
    /* 定义曲面形状 */
        gluNurbsSurface  (theNurb,10,knots,10,knots,5*3,3,
&ctrlpoints[0][0][0],5,5,GL_MAP2_VERTEX_3);
    gluEndSurface(theNurb);
    glPopMatrix();
    glutSwapBuffers();
}
```

4.4 程序绘制

结果如图 1 所示。

（下转第 77 页）

网站开发之 Google 地图的应用

庞国明

摘　要：在自己的网站上嵌入谷歌地图应用，这无疑增强了网站的实用性、方便性。用户打开相关网页后，能从地图中直观查看地理位置等信息。通过谷歌地图应用，用户可以直接在地图上“划区域”、“定位星标点”等编辑操作，网页也可以相应在地图上的点击事件、拖动事件等操作。Google MAp API v3 第三版免费提供了以上服务。

关键词：Javascript 环境；Google MAP API v3 第三版；Web API 项目；Jquery 框架；Html 代码

1　引言

现在流行的最新版正是 Google MAP API v3。Google Maps API 是 Google 为开发者提供的地图编程 API。它允许开发者在不必建立自己的地图服务器的情况下，将 GoogleMaps 地图数据嵌入到网站之中，从而实现嵌入 GoogleMaps 的地图服务应用，并借助 GoogleMaps 的地图数据为用户提供位置服务。

Google Maps API 除了帮助开发者将地图嵌入到 Web 应用中之外，还允许开发者利用 JavaScript 进行应用开发拓展，给地图添加标注和折线及其他地图图层覆盖物，或者响应用户的点击动作。

Google 地图 API 是一种通过 JavaScript 将 Google 地图嵌入到您的网页的 API。该 API 提供了大量实用工具用以处理地图，并通过各种服务向地图添加内容，从而使您能够在您的网站上创建功能全面的地图应用程序。

2　Google JavaScript API 接口

2.1　体系结构

JavaScript 是一种能让网页更加生动活泼的程式语言，也是目前网页中设计中最容易学又最方便的语言。通过 Javascript 可以灵活地操作网页中的 DOM 树对象。并且 JavaScript 支持 2D、3D API，也就是说，通过 JavaScript 可以在网页上划出任何 2D、3D 图形。这就为在网页上实现地图显示功能提供了技术基础。现在 JavaScript 的最新版为 JavaScript 1.8。它提供了更加强大的图形图像处理功能。

2.2　开发环境

操作系统：Microsoft Windows。

浏览器：任何主流浏览器均可。

开发环境：txt 记事本、Dreamweaver CS 等任何文本编辑器。

3 需求分析

3.1　功能

(1) 用户通过 Google MAP API v3 可以直接在网页上看到矢量地图、卫星图像地图以及地图上的各个标记（道路标记、场所标记、建筑物标记、自然环境标记等）（如图 1、图 2 所示）。

图 1

图 2

图 3

(2) 网站可以利用 API 接口，绘制相应地图标记，如区域标记、星标标记，供用户在网页上查看（如图 3 所示）。

(3) 网站可以利用 API 接口，为用户提供在地图上绘制区域标记、星标的功能，并且这所生成的相应数据也应能被网站后台获取并保存（如图 4 所示）。

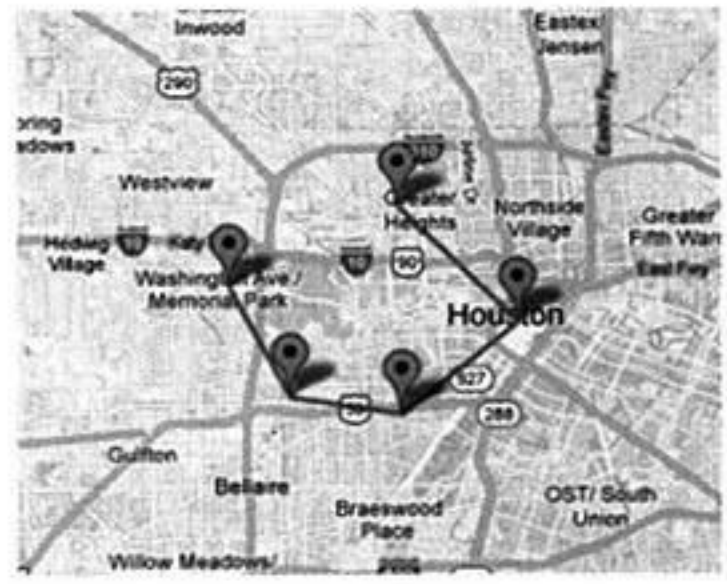


图 4

3.2 性能

Google 地图要在自己的网站上显示自然也要有相应的条件，这里所说的条件就是支持地图功能的底层环境。

对于 Google 地图的应用要求为以下两条：

（1）浏览器必须是 IE6 或其以上版本，因为 JavaScript 是运行在 IE 浏览器中的，而 Google 地图又是 JavaScript 编写脚本的，因此浏览器的功能至关重要。。

（2）由于要调用谷歌提供的服务，所以在应用地图前必须引入谷歌的开发包，基础 JavaScript 包地址为：http://maps.google.com/maps/api/js?sensor=true，如果要在地图上实现一些特别的功能则必须再引入专用包（如实现在地图上做文字标记的 maplabel.js）。

3.3 可靠性与可用性

Google MAP API v3 所兼容的浏览器为：

（1）IE 6.0+ (Windows) 以及 IE 内核的其他浏览器，如：搜狗高速浏览器、遨游浏览器、360 浏览器、世界之窗浏览器等。

（2）Firefox 2.0+ (WindowslMaclLinux)

（3）Safari 3.1+ (MaclWindows)

（4）chrome 谷歌浏览器 (Windows)

（5）opera 10+ (Windows)

由此看见 Google MAP API v3 兼容现在主流的多个版本的浏览器。

4 系统实现

4.1 搭建 Google 地图只用来显示的区域

首先，不讨论地图上的各种功能，先将一张空的地图窗口显示在自己的网页上，这自然是实现各种地图功能的基础和先决条件。

第一步，在 html 头标记<head></head>中引入 Google 地图基本的 JavaScript 包，代码为：

```
<script type ='text/javascript' src ='http://maps.google.com/maps/api/js?sensor=true'></script>
```

第二步，在<body></body>标记内定义地图窗口的 div 层，代码为：

```
<div id='map_canvas' style='width:1200px; height:800px'></div>
```

以上代码在网页内容区域定义了地图要显示的窗口层，以及这个窗口的宽和高。

第三步，在<head></head>标记内，写入 Google MAP API 调用代码：

```
<script type='text/javascript'>
var map; //定义地图对象
function initialize() { //网页打开时将执行此方法
 var latlng = new google.maps.LatLng( 29.7538658334218,-95.3460513679013 //定义地图窗口默认显示的地理位置。
);
 var myOptions = {
  zoom: 13,  //定义地图默认的缩放比例
  center: latlng,
  streetViewControl: false,
  panControl: false,
mapTypeId: google.maps.MapTypeId.ROADMAP };
//定义地图显示模式是路况地图还是卫星地图
});
</script>
```

通过以上代码提供的功能现在就可以在自己的网页上显示一张空地图了，地图默认显示的是美国休斯敦市，地图的默认缩放比例为 13（如图 5、图 6 所示）。

图 5

图 6

4.2 给地图加入星标点

地图的星标点就是用红色小球标记出来的地理位置点，用户可以用此星标点很容易地识别出地理位置点，并且将鼠标移至星标点时会显示星标点的备注名称等信息（如图 7、图 8 所示）。

图 7

图 8

实现代码为根据 Google map api 在：

```
mapTypeId: google.maps.MapTypeId.ROADMAP };
```

之后加入以下代码：

```
map = new google.maps.Map (document.getElementById('map_canvas'), myOptions);
var latlng1 = new google.maps.LatLng ( 29.7501800, -95.3499904 ); //定义星标点的位置
var marker1 = new google.maps.Marker({
position: latlng1,
map: map,
```

```
title:'1. Capitol Oaks' }); //定义星标点的提示文本
google.maps.event.addListener(marker1, "click", function() {
            window.location.href ='cavahome.aspx?idcia =
101&webcia=(101)&wlink=7'
});
```

细心的读者可能已经发现，在这个星标点也也定义了鼠标点击事件的驱动代码，就是说如果鼠标单击此星标点，那么JavaScript 将捕获此事件并作出相应的动作，这就是 google.maps.event.addListener (marker1, " click" , function () 一段所起的功能。

在此有以下几点要说明：

(1) 鼠标单击事件驱动代码里一般都是放入链接代码，当鼠标单击一个星标后，JavaScript 将把页面跳转到“cavahome.aspx”页面。

(2) 除了“click”事件以外，自然还有其他的事件可以获取比如双击事件，鼠标移动上事件等，这些都是可以获取的到的，通过这些事件自然可以实现一些实用的功能。

(3) 如果对星标点的小红球图像感觉不满意的话，也可以根据其 JavaScript API 接口更换其图片，将小红球换成自己想要的图标，代码为在 var marker = new google.maps.Marker 后加入以下代码：

```
icon:'images/marker/marker.png',
```

其效果如图 9 所示，将其换为棕色的星标图片。

图 9

4.3 给地图加入区域标记

为了能够在地图上明显地标记出一片区域并能对此区域进行交互就需要 Google Map API 里面的“polygon”对象，“polygon”对象可以在地图指定的区域画出多边形区域，并且可以设置区域上的文字标记，点击事件等。

要实现在地图上显示多边形区域只要在添加星标的代码后加入以下代码：

```
var county;
 var fipsCoords;
 fipsCoords = [ //定义多边形各个定点坐标
    new google.maps.LatLng (29.77181441550003, -
95.32875395974122)
    , new google.maps.LatLng (29.76019144780945, -
95.35038329323731)
    , new google.maps.LatLng (29.745139191510482, -
95.3636012192627)
    , new google.maps.LatLng (29.741934706562126, -
95.36385871132813)
    , new google.maps.LatLng (29.739773484473744, -
95.36188460549317)
    , new google.maps.LatLng (29.738879172054677, -
95.3588805313965)
    , new google.maps.LatLng (29.74066778891791, -
95.35553313454591)
    , new google.maps.LatLng (29.73559995829096, -
95.34780837258302)
    , new google.maps.LatLng (29.754975575321616, -
95.33510543068848)
    , new google.maps.LatLng (29.76168164722902, -
95.3273806687256)
    , new google.maps.LatLng (29.767493213197202, -
95.32789565285645)
  ];
  county = new google.maps.Polygon({
  paths: fipsCoords,
  strokeColor: '#006778', //设置多边形边界线区域颜色
  strokeOpacity: 0.8, //设置多边形边界不透明度
   strokeWeight: 2, //设置多边形边界线粗度
   fillColor: '#006778', //设置多边形填充颜色
  fillOpacity: 0.35 //设置多边形内部透明度
  });
   county.setMap(map);
google.maps.event.addListener(county, 'click', //设置多边形
//区域上的单击事件
function(event) {
   location.href = 'FindAHome.aspx?idcia=101&idarea=10';
  });
   var mapLabel = new MapLabel({
    text: 'Downtown', //设置多边形区域提示文本
    position: new google.maps.LatLng (29.7538658334218,-
95.3460513679013//提示文本位置
   ),
    map: map,
    fontSize: 30, //设置文字大小
    align: 'center' //文字对齐方式
   });
    mapLabel.set ('position', new google.maps.LatLng
(29.7538658334218,-95.3460513679013
  ));
```

代码开头首先定义了一个“fipsCoords”数组，此数组保存了多边形各个定点的地理位置坐标值，其中每个坐标有一个 google.maps.LatLng 对象。

之后定义 google.maps.Polygon 对象，这正是构建多边形的

"polygon" 对象，此对象初始化时就要设定好多边形边界线区域颜色、不透明度、边界线粗度、内部透明度、填充颜色。之后用此对象的.setMap (map) 方法将此多边形应用到地图中去。

为了在多边形上显示提示性的文本，在代码中又定义了"MapLabel" 对象。此对象初始化时要求给予文本位置、字体大小、对齐方式、文字内容等参数。之后用此对象的 map: map 参数将此文本对象应用于地图上。

细心的读者又会发现 google.maps.event.addListener 方法，不错此方法正是该多边形的单击事件响应方法，当鼠标在多边形上单击时就会触发此方法，从而去执行跳转代码 "location.href = 'FindAHome.aspx?idcia=101&idarea=10';"。

4.4 可编辑多边形区域

在地图应用中除了能从网站后台初始化多边形区域外，Google Map API 还提供了让用户在浏览器端编辑多边形的区域的功能，并且编辑完成的多边形区域数据可以通过 Javascript 提取出来并反馈给后台处理。

第一步，应当引入 Google map api 的响应借口和 jquery 基础包，代码如下：

```
<script type='text/javascript' src='http://maps.google.com/maps/api/js?sensor=true'></script>
  <script src='js/jquery-1.7.2.min.js' type='text/javascript'></script>
   <script  src ='js/polygon.min.js'  type ='text/javascript' ></script>
```

第二步，引入以上这些基础包后就可以调用 Google Map API 接口了，首先要在<body></body>标记中加入地图框体代码： ，此代码定义了地图在网页中显示的位置大小。

第三步，用 JavaScript 调用 Google Map API 的接口，初始化地图。代码如下：

```
<script type='text/javascript'>
$(function(){ //页面加载时的 load 事件
      var  latlng  =  new  google.maps.LatLng  ( 29.751026234150952,-95.40917731484376 ); //定义地图默认
//显示的地理位置
    var myOptions = {
      zoom: 11, //定义缩放值
      center: latlng,
      mapTypeId: google.maps.MapTypeId.ROADMAP  };
       var  map  =  new  google.maps.Map  (document.getElementById('map_canvas'), myOptions); //设置地图所在
//的 div
  });
</script>
```

通过以上代码就可以在网页上绘制多边形区域了，如图 10、图 11 所示。

图 10

图 11

以上代码只实现了绘制多边形的功能，那怎样提取绘制多边形后生成的地图数据呢？下面通过以下方式获取所生成的数据：

第一步，在<body></body>标签内添加按钮和 div 的 html 代码：

```
<input id='reset' value='Reset' type='button' />
 <input id='showData'  value='Show Paths' type='button' />
  <div id='dataPanel'>  </div>。
<div id='dataPane2' />
```

在此定义了一个输出数据按钮和一个重置按钮。

第二步，在 javascript 调用 Google map api 的代码后加入以下代码：

```
var creator = new PolygonCreator(map);
$('#reset').click(function(){
 $('#dataPanel').empty();
 creator.destroy();
 creator=null;
 creator=new PolygonCreator(map);
});
$('#showData').click(function(){ //显示数据按钮事件
 $('#dataPanel').empty();
 $('#dataPane2').empty();
 if(null==creator.showData()){
  $('#dataPanel').append('Please first create a polygon');
 }else{
  $('#dataPane2').append(map.getZoom());
  $('#dataPanel').append(creator.showData());
 } });
$('#showColor').click(function(){//显示颜色按钮事件
 $('#dataPanel').empty();
 if(null==creator.showData()){
  $('#dataPanel').append('Please first create a polygon');
 }else{
  $('#dataPanel').append(creator.showColor());
 }
});
```

这样就写完了所有的关于编辑地图并显示相应数据的代码了，然后试一下效果。

（下转第 90 页）

网络五子棋的通信原理及编程

凌仕华　汪 琴

摘　要：五子棋是一种广为人知的益智娱乐游戏，当今计算机网络应用几乎无处不在。论述了通过计算机网络运行五子棋的原理及Java语言关键代码实现。

关键词：计算机网络；五子棋；Java语言；Socket通信

五子棋是一种广为人知的益智娱乐游戏，现在的很多家庭有两台及以上的计算机组成局域网，这样可以让五子棋能通过网络来博弈。下文就如何通过局域网或因特网互连来博弈的网络五子棋实现进行讲解，程序代码采用Java语言编写。

1　原理

计算机中的网络应用程序通过网络（包括局域网、因特网等）将数据发给对方应用程序，这种数据传输是通过计算机网络的应用层中的某个端口来实现。也即是服务器端应用程序启动系统的某个端口（例如SQL Server的1433端口），然后不间断地侦听该端口传来的数据并做相应处理或将数据从某端口发送给对方。在Java语言中，首先是通过ServerSocket类来创建一个服务器套接字（Server Socket），再在该套接字上绑定一个端口。

例如：

```
ServerSocket chatSocketServer = new ServerSocket(8500);
    // 创建一个服务器套接字(Socket),并绑定到 8500 端口
//(port)
```

接着通过创建的ServerSocket对象中的accept ()方法来创建一个套接字（Socket），然后再在创建的套接字端口上侦听从网络传来的数据，例如：

```
Socket serverSocket = chatSocketServer.accept();
```

在这Java语言中accept ()方法是用来侦听（listen）从某个连接套接字端口上是不是有数据传来，如果有就接收该数据，否则就不断侦听。如果要往套接字中接入的某个连接传输数据给对方主机应用程序，只要通过套接字（在Java中是Socket类的对象）中的getOutputStream ()方法创建一个输入输出流，然后再往其中写入想要输入的数据。程序如下：

```
String host = serverSocket.ge:InetAddress().getHost
Name(); // 获取对方主机信息
String ip = serverSocket.getInetAddress().getHost
Address(); // 获取对方 IP 地址
    // 获取网络输入流,然后就可以往里面写入数据并发送给
//对方
InputStream  is = serverSocket.getInputStream();
```

网络五子棋程序采用C/S模式，该程序既是客户端也是服务器端。五子棋实现的原理是创建一个字节二维数组，数组中存放棋子（白方用1代表，黑方用-1代表，没有棋子则为0），然后通过网络将该数组发给彼此，如果数组中有数据改变（从0变为1或-1）则立刻更新屏幕，就像用户下了一颗棋一样，如果数组矩阵中有某颜色五个棋子并列排成一行、一列或一斜列都算是赢了：

```
private byte[][] chessmanArray = new byte[15][15]; // 定
//义棋子数组,有 15 行与 15 列矩阵
```

2　编程

界面如图1所示。

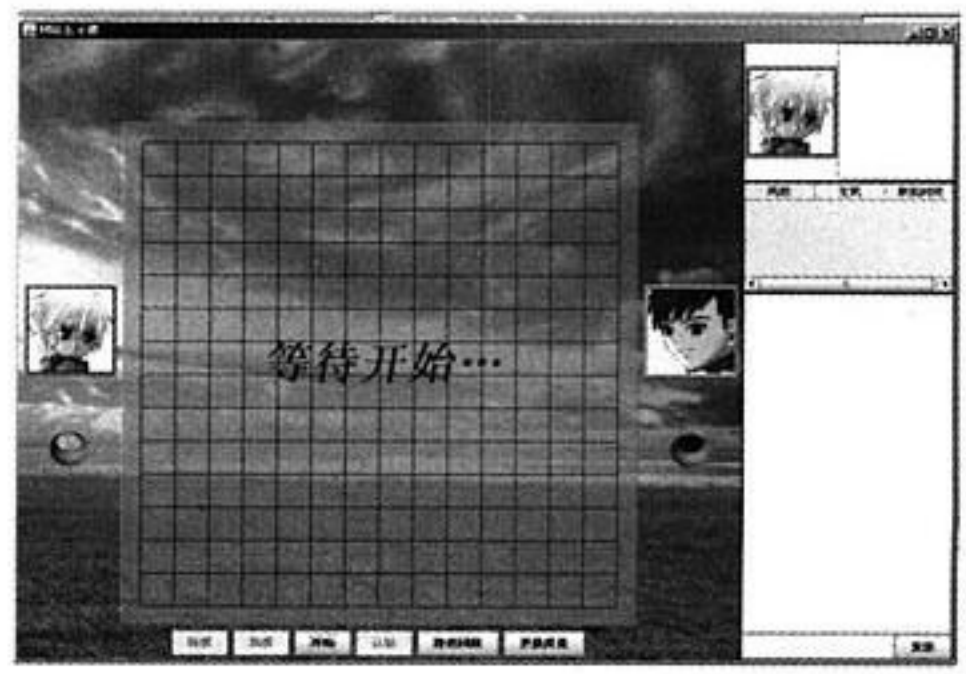


图1　程序运行界面

2.1　在屏幕中绘制棋盘

首先在屏幕中制作一个棋盘，通过java.awt中Graphics2D类的对象中的方法来实现。程序如下：

```
public class ChessBoardPanel extends javax.swing.JPanel {
//代表棋盘面板
```

在ChessBoardPanel类里面包括绘制棋盘的drawPanel ()方法，程序如下：

```
private void drawPanel(Graphics2D g) {
```

```
        Composite composite = g.getComposite(); // 备份
//合成规则
        Color color = g.getColor(); // 备份前景颜色
        g.setComposite    (AlphaComposite.SrcOver.derive
(0.6f));// 设置透明合成
    g.setColor(new Color(0xAABBAA)); // 设置前景白色
        g.fill3DRect(0, 0, getWidth(), getHeight(), true);
// 绘制半透明的矩形
        g.setComposite(composite); // 恢复合成规则
        g.setColor(color); // 恢复原来前景色
    int w = getWidth(); // 棋盘宽度,从 JPanel 继承而来
    int h = getHeight(); // 棋盘高度,从 JPanel 继承而来
        int chessW = w / 15, chessH = h / 15; // 棋子宽度
//和高度
    int left = chessW / 2 + (w % 15) / 2; // 棋盘左边界
        int right = left + chessW * 14; // 棋盘右边界
        int top = chessH / 2 + (h % 15) / 2; // 棋盘上边界
        int bottom = top + chessH * 14; // 棋盘下边界
        for (int i = 0; i < 15; i++) {
            // 画每条横线
            g.drawLine(left, top + (i * chessH), right, top
+ (i * chessH));
        }
        for (int i = 0; i < 15; i++) {
            // 画每条竖线
            g.drawLine(left + (i * chessW), top, left + (i *
chessW), bottom);
        }
    }
```

2.2 用户鼠标单击棋盘下棋

当用户用鼠标在屏幕上点击，则在棋盘中鼠标单击处下一颗棋子，程序如下（程序同在 ChessBoardPanel 类中）：

```
private void formMouseClicked (java.awt.event.MouseEvent
evt) {
        // 判断是否开始游戏
    if (! start || ! isTowardsStart() || myColor == 0 || ! turn) {
            return;
}
//start 变量代表下棋是否已开始,isTowardsStart()代表对方是
//不是也开始
  //myColor 是代表棋子的颜色,颜色包括 1(白色)和-1(黑色),
//turn 是不是轮到某方
        Point point = evt.getPoint();//取得用户鼠标点击屏
//幕中点的坐标,判断是那个格子
         int xindex = point.x / chessWidth;//除以棋子的宽
//度,取得格子 X 坐标
        int yindex = point.y / chessHeight; //除以棋子的高
//度,取得格子 Y 坐标
        byte [] [] chessmanArray = chessModel1.
getChessmanArray();//取得代表棋盘数组
        if (chessmanArray[xindex][yindex] == 0) {
            turn = ! turn;
        //棋子最近颜色,区别已经下的和刚下的棋子
        chessmanArray[xindex][yindex] = (byte) (myColor ^ 3);
        //更新棋盘数据,从而更新屏幕
            chessModel1.setChessmanArray
(chessmanArray);
            chessPanel.backButton.setEnabled(false);
        //整理棋盘,把前一步对方刚下棋子变为普通棋子
        zhengliBoard((byte) -myColor);
        //判断是不是五个同颜色棋子排成一行、一列或一斜列
        int winColor = arithmetic(myColor, xindex, yindex);
        //发送数据给对方主机五子棋应用程序
      chessPanel.send(chessModel1.getChessmanArrayCopy());
            if (winColor ! = 0 && winColor == myColor) {
       chessPanel.send(ChessPanel.WIN); // 发送胜利代码
                win = true;
                chessPanel.reInit();
            }// 在判断胜负情况后再发送 model 中的棋盘
//数组,因为这个数组可能带有标
    //识连线的棋子数据
        }
    }
```

整理棋盘方法代码如下：

```
    /* 整理棋盘,将标识上一步的棋子恢复普通棋子 */
    public void zhengliBoard(byte color) {
        byte [] [] chessmanArray = chessModel1.
getChessmanArray();
        for (int i = 0; i < chessmanArray.length; i++) {
    for (int j = 0; j < chessmanArray[i].length; j++) {
                if (chessmanArray[i][j] == (color ^ 3)) {
    chessmanArray[i][j] = color;//改变为原来颜色棋子
                }
            }
        }
        chessModel1.updateChessmanArray
(chessmanArray);//更新棋盘数组
        repaint();//重绘棋盘
    }
```

2.3 绘制刚下的棋子

绘制刚下的棋子代码如下：

```
  public void paint(Graphics g1) {
        Graphics2D g = (Graphics2D) g1;
        super.paint(g); // 调用父类的绘图方法
        if (chessPanel ! = null) {
            chessPanel.setTurn(turn);
        }
         Composite composite = g.getComposite(); // 备份
//合成模式
        drawPanel(g); // 调用绘制棋盘的方法
        g.translate(4, 4);
        size = new Dimension(getWidth(), getHeight());
```

GAME PROGRAM

```
// 获取棋盘面板的大小
        chessWidth = size.width / 15; // 初始化棋子宽
        chessHeight = size.height / 15; // 初始化棋子高
        byte [] [] chessmanArray = chessModel1.
getChessmanArrayCopy();//复制棋盘数组数据
        for (int i = 0; i < chessmanArray.length; i++) {// 遍
//历棋盘数据模型绘制棋子
        for (int j = 0; j < chessmanArray[i].length; j++) {
                byte chessman = chessmanArray[i][j];
                int x = i * chessWidth;
                int y = j * chessHeight;
                if (chessman ! = 0)
            System.out.println("chess is:" + chessman);
    if (chessman == WHITE_CHESSMAN) {// 绘制白棋
                        g.drawImage
(white_chessman_img, x, y, chessWidth,chessHeight, this);
                        } else if (chessman == BLACK_
CHESSMAN) {// 绘制黑棋
                        g.drawImage
(black_chessman_img, x, y, chessWidth,chessHeight, this);
                        } else if (chessman == (WHITE_
CHESSMAN ^ 3)) { // 绘制最近的白棋落子
                        g.drawImage
(white_chessman_img, x, y, chessWidth,chessHeight, this);
                g.drawRect(x, y, chessWidth, chessHeight);
                        } else if (chessman == (BLACK_
CHESSMAN ^ 3)) { // 绘制最近的黑棋落子
                        g.drawImage
(black_chessman_img, x, y, chessWidth,chessHeight, this);
                g.drawRect(x, y, chessWidth, chessHeight);
                        } else if (chessman == ((byte) (WHITE_
CHESSMAN ^ 8))) {
// 绘制导致胜利的连线白棋
                        g.drawImage
(white_chessman_img, x, y, chessWidth,chessHeight, this);
                        g.drawImage (rightTop_img, x, y,
chessWidth, chessHeight,this);
                        } else if (chessman == (BLACK_
CHESSMAN ^ 8)) {
    // 绘制导致胜利的连线黑棋
                        g.drawImage
(black_chessman_img, x, y, chessWidth,chessHeight, this);
                        g.drawImage (rightTop_img, x, y,
chessWidth, chessHeight,this);
                }
            }
        }
        if (! isStart()) { // 如果游戏不处于开始状态
            if (towardsWin || win || draw) {
        // 如果游戏处于胜利或和棋状态,绘制棋盘提示信息
    // 设置 70%透明的合成规则
                        g.setComposite (AlphaComposite.
SrcOver.derive(0.7f));
String mess = "对方胜利"; // 定义提示信息
                g.setColor(Color.RED); // 设置前景色为红色
                    if (win) { // 如果是自己胜利
                mess = "你胜利了"; // 设置胜利提示信息
                        g.setColor (new Color(0x007700));
// 设置绿色前景色
                    } else if (draw) { // 如果是和棋状态
                mess = "此战平局"; // 定义和棋提示信息
                        g.setColor(Color.YELLOW); // 设置
//和棋信息使用黄色提示
                    }
            // 设置提示文本的字体为隶书、粗斜体、大小 72
                    Font font = new Font (" 隶书", Font.
ITALIC | Font.BOLD, 72);
                    g.setFont(font);
                    // 获取字体渲染上下文对象
                    FontRenderContext context = g.getFont
RenderContext();
            // 计算提示信息的文本所占用的像素空间
                        Rectangle2D stringBounds = font.
getStringBounds(mess, context);
                        double fontWidth = stringBounds.
getWidth(); // 获取提示文本的宽度
    // 居中绘制提示信息
                    g.drawString (mess, (int) ((getWidth () -
fontWidth) / 2),getHeight() / 2); g.setComposite (composite);
// 恢复原有合成规则
    } else { // 如果当前处于其他未开始游戏的状态
            String mess = "等待开始…"; // 定义等他提示信息
                    Font font = new Font (" 隶书", Font.
ITALIC | Font.BOLD, 48);
                    g.setFont(font); // 设置 48 号隶书字体
                    FontRenderContext context = g.
getFontRenderContext();
                    Rectangle2D stringBounds = font.
getStringBounds(mess, context);
                    double fontWidth = stringBounds.
getWidth(); // 获取提示文本的宽度
    // 居中绘制提示文本
                    g.drawString (mess, (int) ((getWidth () -
fontWidth) / 2),getHeight() / 2);
                }
        }
```

2.4 五子棋算法

判断棋子是否形成 5 个一行、一列或一排，程序代码如下：

```
/**
 * 五子棋算法
 * @param n      – 代表棋子颜色的整数
 * @param Arow   – 行编号
 * @param Acolumn – 列编号
```

```
     * @return 胜利一方的棋子颜色的整数
     */
    public int arithmetic(int n, int Arow, int Acolumn) {
        int n3 = n ^ 3;
        byte n8 = (byte) (n ^ 8);
                        byte [][] note = chessModel1.
getChessmanArrayCopy();//复制棋盘数据
        int BCount = 1;
        // 纵向查找
        boolean Lbol = true;
        boolean Rbol = true;
        BCount = 1;
        for (int i = 1; i <= 5; i++) {
        if ((Acolumn + i) > 14) {// 如果棋子超出最大列数
                Rbol = false;
            }
        if ((Acolumn - i) < 0) {// 如果棋子超出最小列数
                Lbol = false;
            }
            if (Rbol == true) {
                if (note[Arow][Acolumn + i] == n
    || note[Arow][Acolumn + i] == n3) {// 如果横向向右有相
//同的棋子
                    ++BCount;
                    note[Arow][Acolumn + i] = n8;
                } else {
                    Rbol = false;
                }
            }
            if (Lbol == true) {
                if (note[Arow][Acolumn - i] == n
                        || note [Arow] [Acolumn - i]
== n3) {// 如果横向向左有相同棋子
                    ++BCount;
                    note[Arow][Acolumn - i] = n8;
                } else {
                    Lbol = false;
                }
            }
    if (BCount >= 5) {// 如果同类型的棋子数大于等于 5 个
                note[Arow][Acolumn] = n8;
chessModel1.updateChessmanArray(note);
                repaint();
                return n; // 返回胜利一方的棋子
        }
}
// 横向查找
note = chessModel1.getChessmanArrayCopy();
boolean Ubol = true;
boolean Dbol = true;
BCount = 1;
for (int i = 1; i <= 5; i++) {
if ((Arow + i) > 14) {// 如果超出棋盘的最大行数
            Dbol = false;
        }
if ((Arow - i) < 0) {// 如果超出棋盘的最小行数
            Ubol = false;
        }
        if (Dbol == true) {
            if (note[Arow + i][Acolumn] == n
                    || note [Arow + i] [Acolumn]
== n3) { // 如果向上有同类型的棋子
                ++BCount;
                note[Arow + i][Acolumn] = n8;
            } else {
                Dbol = false;
            }
        }
        if (Ubol == true) {
            if (note[Arow - i][Acolumn] == n
                    || note [Arow - i] [Acolumn]
== n3) { // 如果向下有同类型的棋子
                ++BCount;
                note[Arow - i][Acolumn] = n8;
            } else {
                Ubol = false;
            }
        }
    if (BCount >= 5) { // 如果同类型的棋子大于等于 5 个
                note[Arow][Acolumn] = n8;
        chessModel1.updateChessmanArray(note);
                repaint();
                return n; // 返回胜利一方的棋子
        }
    }
    // 正斜查找
    note = chessModel1.getChessmanArrayCopy();
    boolean LUbol = true;
    boolean RDbol = true;
    BCount = 1;
    for (int i = 1; i <= 5; i++) {
        if ((Arow - i) < 0 || (Acolumn - i < 0)) {// 如果
//超出左面的斜线
            LUbol = false;
        }
        if ((Arow + i) > 14 || (Acolumn + i > 14)) {
//如果超出右面的斜线
            RDbol = false;
        }
        if (LUbol == true) {
    // 如果左上斜线上有相同类型的棋子
        if (note[Arow - i][Acolumn - i] == n || note[Arow
- i][Acolumn - i] == n3) { ++BCount;
                note[Arow - i][Acolumn - i] = n8;
```

```
                } else {
                    LUbol = false;
                }
            }
            if (RDbol == true) {
    // 如果右下斜线上有相同类型的棋子
                if  (note [Arow + i][Acolumn + i] == n||
note[Arow + i][Acolumn + i] == n3) {
                    ++BCount;
                    note[Arow + i][Acolumn + i] = n8;
                } else {
                    RDbol = false;
                }
            }
if (BCount >= 5) {// 如果同类型的棋子大于等于 5 个
                note[Arow][Acolumn] = n8;
        chessModel1.updateChessmanArray(note);
                repaint();
                return n; // 返回胜利一方的棋子
            }
        }
        // 反斜查找
        note = chessModel1.getChessmanArrayCopy();
        boolean RUbol = true;
        boolean LDbol = true;
        BCount = 1;
        for (int i = 1; i <= 5; i++) {
            if ((Arow - i) < 0 || (Acolumn + i > 14)) {
                RUbol = false;
            }
            if ((Arow + i) > 14 || (Acolumn - i < 0)) {
                LDbol = false;
            }
            if (RUbol == true) {
    // 如果左下斜线上有相同类型的棋子
                if  (note [Arow - i][Acolumn + i] == n||
note[Arow - i][Acolumn + i] == n3) {
                    ++BCount;
                    note[Arow - i][Acolumn + i] = n8;
                } else {
                    RUbol = false;
                }
            }
            if (LDbol == true) {
    // 如果右上斜线上有相同类型的棋子
                if  (note [Arow + i][Acolumn - i] == n||
note[Arow + i][Acolumn - i] == n3) {
                    ++BCount;
                    note[Arow + i][Acolumn - i] = n8;
                } else {
                    LDbol = false;
                }
            }
        if (BCount >= 5) {// 如果同类型的棋子大于等于 5 个
                    note[Arow][Acolumn] = n8;
                    chessModel1.updateChessmanArray
(note);//更新棋盘模型
                    repaint();
                    return n;// 返回胜利一方的棋子
                }
            }
        return 0;
    }
```

2.5 服务器接收与发送数据

在程序的主界面，也即是 main () 函数所在类，其中包括初始化创建服务器并打开相应端口侦听相关代码，还有主界面的创建代码，程序部分如下：

```
public class MainFrame extends javax.swing.JFrame {
    private ObjectOutputStream objout;//从套接字中取得的
//输入与输出流
    Socket serverSocket; //服务器端套接字
    public MainFrame() {
        initComponents(); // 初始化主窗体界面(如图 1 所
//示)
        setGlassPane(loginPanel1); // 设置登录面板为玻璃
//面板,如图 2 所示
        loginPanel1.setVisible(true); // 显示登录面板,用户
//输入昵称与要连接的 IP 地址
    }
```

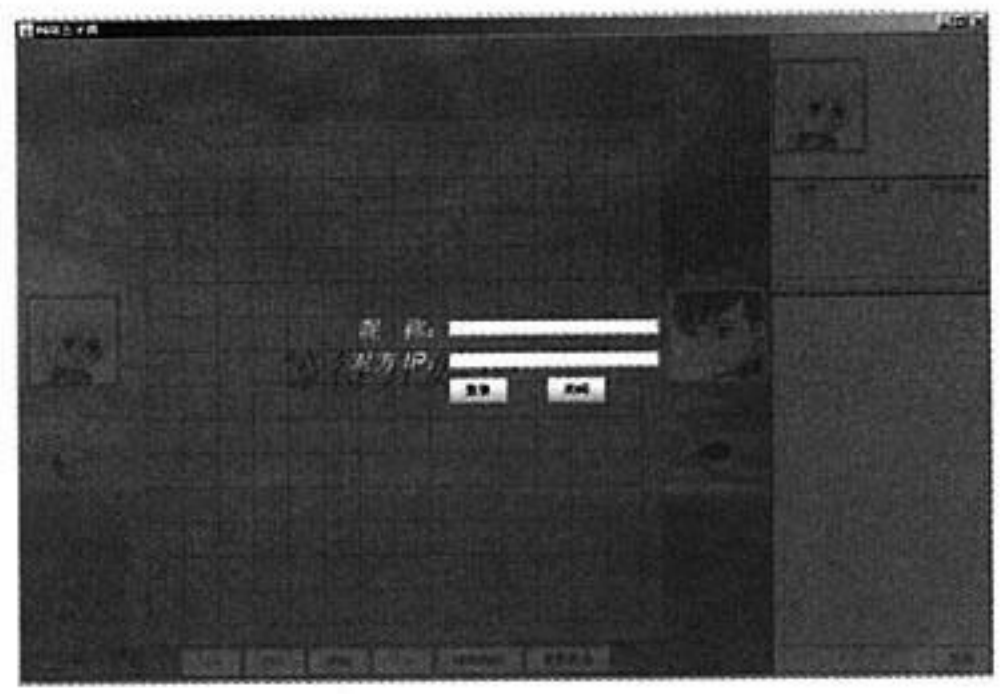


图 2 登录界面

创建服务器套接字与端口：

```
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater (new Runnable()
{
        public void run() {
            try {
MainFrame frame = new MainFrame();
                frame.startServer();
                frame.setVisible(true);
```

```
    /** 在指定端口启动 Socket 服务器 */
    public void startServer() {
        try {
            // 创建 Socket 服务器对象
            final ServerSocket chatSocketServer = new
ServerSocket(8500);
            // 创建接收信息的线程, 8500 为端口
            new ReceiveThread (chatSocketServer, this).
start();//启动线程
        }
```

从服务器接收客户端数据或发送数据给客户端程序在ReceiveThread类中，程序代码如下：

```
public class ReceiveThread extends Thread {
    private final ServerSocket chatSocketServer;//服务器端
//Socket 服务器
    MainFrame frame;
    private String host;//客户端主机信息
public void run() {
        while (true) {//无限服务连接上的请求
            try {
                frame.serverSocket = chatSocket
Server.accept(); // 接收 Socket 连接
            Socket serverSocket = frame.serverSocket;
                host = serverSocket.getInetAddress ().
getHostName(); // 获取对方主机信息
                String ip = serverSocket.getInetAddress
().getHostAddress(); // 获取对方 IP
                int link = JOptionPane.showConfirm
Dialog(frame, "收到" + host
                        + "的联机请求,是否接受? ");
// 询问是否接受联机
                if (link == JOptionPane.YES_OPTION) {
// 如果接受联机
                LoginPanel loginPanel = (LoginPanel)
frame.getRootPane()
                        .getGlassPane(); // 获取
//登录面板的实例
                loginPanel.setLinkIp(ip); // 设置登
//录面板的对家 IP 信息
                }
                serverSocket.setOOBInline(true); // 启用
//紧急数据的接收
                InputStream is = serverSocket.
getInputStream(); // 获取网络输入流
                ObjectInputStream objis = new
ObjectInputStream(is);// 创建对象输入流
                while (frame.isVisible()) {
    serverSocket.sendUrgentData(255); // 发送紧急数据
    // 从对象输入流读取 Java 对象
        Object messageObj = objis.readObject();
                if (messageObj instanceof String) {
// 如果读取的对象是 String 类型
  String name = frame.getTowardsUser().getName();// 获取对
//家昵称将字符串信息添加到通讯面板
    frame.appendMessage(name + ":" + messageObj);
    // 如果读取的是字节数组对象
        } else if (messageObj instanceof byte[][]) {
    // 将数组对象设置为棋盘模型数据
ChessModel.getInstance ().setChessmanArray ((byte [] [])
messageObj);
    // 获得走棋权限
    frame.getChessPanel1().getChessBoardPanel1().setTurn
(true);
                byte myColor = frame.getChessPanel1
().getChessBoardPanel1()
                        .getMyColor();
// 获取自己的棋子颜色
                frame.getChessPanel1 ().
getChessBoardPanel1().zhengliBoard(
                        myColor); // 整理棋盘
                frame.getChessPanel1().backButton.set
Enabled(true);//悔棋按钮可用
                } else if (messageObj instanceof
Integer) {// 如果是整形对象
                oprationHandler(messageObj);
// 命令代码的接收和处理方法
                } else if (messageObj instanceof
UserBean) {// 如果是用户实体对象
        UserBean user = (UserBean) messageObj;
                frame.setTowardsUser (user);
// 设置对家信息
                }
            }
```

2.6 承载棋盘棋子的数据模型 JavaBean

该数据模型中有个很重要的作用是当代表棋盘的二维数组中的数据改变时（例如当用户单击棋盘下一颗棋子），该模型会立刻触发一个 PropertyChangeSupport 中安装的侦听器，从而重绘屏幕棋盘。程序代码如下：

```
public class ChessModel extends Object implements
Serializable {
    private PropertyChangeSupport propertySupport; // 定义
//属性工具类
    private static ChessModel model; // 定义自身的变量
    private byte[][] chessmanArray = new byte[15][15]; // 定
//义棋子数组
    public static final String PROP_CHESSMANARRAY = "
chessmanArray"; // 定义属性名称
    public static ChessModel getInstance() {
        if (model == null) {
            model = new ChessModel();
        }
```

```
        return model;
    }
    public ChessModel() {
        propertySupport = new PropertyChangeSupport
(this);// 初始化属性工具栏
        model = this;
    }
    /**
     * 获取棋盘的棋子数组的方法
     * @return – 代表棋子的数组
     */
    public byte[][] getChessmanArray() {
        return chessmanArray; // 返回棋子数组
    }
    /**
     * 设置棋子数组的方法
     * @param chessmanArray – 一个代表棋盘棋子的二维
数组
     */
    public void setChessmanArray(byte[][] chessmanArray) {
        this.chessmanArray = chessmanArray;
        propertySupport.firePropertyChange
(PROP_CHESSMANARRAY, null,
                chessmanArray); // 报告所有已注册侦听
//器的绑定属性更新
    }
    /**
     * 更新棋子数组的方法,不会产生跟新事件
     * @param chessmanArray
     */
    public synchronized void updateChessmanArray(byte[][]
chessmanArray) {
        this.chessmanArray = chessmanArray;
    }
    /**
     * 添加事件监听器的方法
     * @param listener – 事件监听器
     */
                public void addPropertyChangeListener
(PropertyChangeListener listener) {
        propertySupport.addPropertyChangeListener
(listener); // 添加事件监听器
    }
    /**
     * 移除事件监听器的方法
     * @param listener – 事件监听器
     */
                public void removePropertyChangeListener
(PropertyChangeListener listener) {
        propertySupport.removePropertyChangeListener
(listener); // 移除事件监听器
    }
    /**
     * 获取棋盘上棋子数组的拷贝
     * @return – 棋子数组
     */
    byte[][] getChessmanArrayCopy() {
        byte[][] newArray = new byte[15][15]; // 创建一个
//二维数组
        for (int i = 0; i < newArray.length; i++) {
            // 复制数组
            newArray[i] = Arrays.copyOf(chessmanArray
[i], newArray[i].length);
        }
        return newArray;
    }
}
```

3 结语

网络五子棋是通过局域网或因特网来进行博弈，它既有娱乐性也有益智性。探讨了网络程序通信及五子棋算法相关原理，同时利用该原理用 Java 语言编写了该应用程序。

本程序在 Eclipse JUNO 版本下调试通过。

参考文献

[1] 昊斯特曼. JAVA 核心技术. 北京：机械工业出版社，2008.

（收稿日期：2012-08-13）

（上接第 66 页）

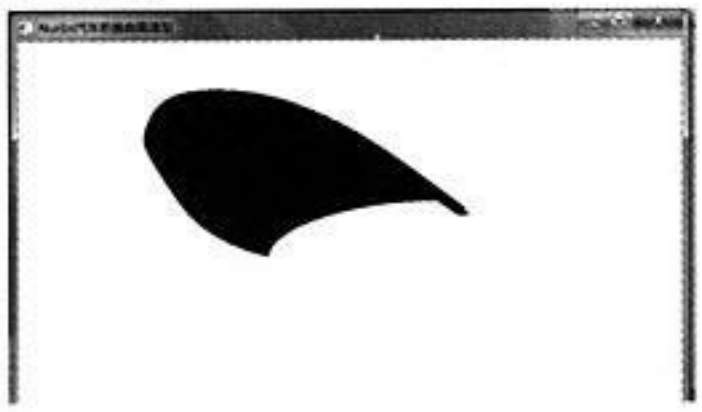


图 1 汽车前盖曲面造型效果图

5 结语

利用 VC++平台与 OpenGL 结合来进行 NURBS 的汽车前盖的曲面造型。结果表明，采用 NURBS 方法来造型汽车前盖曲面，只需要很少量的控制点便可绘制出曲面，不但可以立体显示造型曲面，而且可以全方位观察汽车前盖曲面，同时 OpenGL 技术具有开放性和交互性，可为进一步的开发打下基础和留下空间，为深入研究和研发的人员打下基础，方便继续研究。

（收稿日期：2012-08-16）

Linux 开发归档程序

江 洪

摘 要：在 Linux 图形模式下，开发了一个可以完成 TAR 加入和解出文件及目录功能的程序。程序用到了 Qt 图形界面开发技术，是一个简单实用的例程。

关键词：Linux 系统；归档；TAR 文件；Qt 工具

1 引言

Linux 系统日益受到大家的喜爱，在 Linux 系统下进行程序开发也变得愈发时尚。因此在 Linux 图形界面下开发了一个归档程序，综合运用了 Qt 界面开发技术，结合 Linux 下常用的 TAR 文件结构进行编程。程序在 Fedora13 系统下测试通过，是一个很好的开发实例。

2 Qt 简介

Qt 是一个跨平台的图形界面开发工具。其中拥有大量的类、函数和数据结构，是开发 Linux 图形界面的一个很好的工具。Qt 是基于 C++代码用于 C++开发的程序。

2.1 主窗口

```
QWidget *window;
window=new QWidget;
window->setWindowTitle(str.fromLocal8Bit("TAR 管理"));
window->resize(800,600);
window->show();
```

建立窗口 window；setWindowTitle 函数设置窗口标题；resize 函数设置窗口大小；show 函数显示窗口。

2.2 菜单

```
QMenuBar *menubar;
QMenu *menu1;
QAction *menu1_1,*menu1_2,*menu1_3;
menubar=new QMenuBar(window);
menu1=new QMenu(str.fromLocal8Bit("文件"),menubar);
menubar->addMenu(menu1);
menu1_1=menu1->addAction(str.fromLocal8Bit("加入"));
menu1_2=menu1->addAction(str.fromLocal8Bit("解出"));
menu1->addSeparator();
menu1_3=menu1->addAction(str.fromLocal8Bit("退出"));
connect(menu1_1,SIGNAL(triggered()),SLOT(exec_1_1()));
connect(menu1_2,SIGNAL(triggered()),SLOT(exec_1_2()));
connect(menu1_3,SIGNAL(triggered()),SLOT(exec_1_3()));
```

首先建立菜单条 menubar；再建立菜单 menu1，将 menu1 加入 menubar 中；建立菜单项 menu1_1,menu1-2,menu1_3，将菜单项加入 menu1 中；调用函数 connect 增加菜单项处理函数。

2.3 基本控件

```
QLabel *label1; //标签控件
        QPushButton  *button1,*button2,*button3,*button4,
*button5; //按钮
label1=new QLabel(dlg1); //建立标签
label1->setText(filename); //设置文字
label1->setGeometry(QRect(20,20,300,30)); //设置位置
button1=new QPushButton(str.fromLocal8Bit("确定"),dlg1);
//建立按钮
button1->setGeometry(QRect(350,480,60,25)); //设置按钮
//位置
connect(button1,SIGNAL(clicked()),SLOT(exec_button1_1()));
//增加处理函数
```

2.4 文件对话框

```
//保存文件对话框
filename=QFileDialog::getSaveFileName(
        this,str.fromLocal8Bit("保存"),QDir::currentPath(),
        "TAR files (*.tar);;All files(*.*)");
//打开文件对话框
filename=QFileDialog::getOpenFileName(
        this,str.fromLocal8Bit("打开"),QDir::currentPath(),
        "TAR files (*.tar);;All files(*.*)");
```

2.5 表格控件

```
QTableWidget *table1,*table2; //表格
table1=new QTableWidget(dlg1); //建立表格控件
table1->setRowCount(0); //设置行数
table1->setColumnCount(5);  //设置列数
header<<str.fromLocal8Bit("") //定义类头
    <<str.fromLocal8Bit("名称")<<str.fromLocal8Bit("类型")
    <<str.fromLocal8Bit("大小")<<str.fromLocal8Bit("修改时
间");
table1->setHorizontalHeaderLabels(header); //设置列头
table1->setEditTriggers(QAbstractItemView::NoEditTriggers);
//设置只读
```

```
table1 ->setSel ectionBehavior (QAbstractItemView::
SelectRows); //设置整行选择
table1->setSelectionMode(QAbstractItemView::MultiSelecti
on); //设置多行选择
table1->setColumnWidth(0,20); //设置列宽
table1->setColumnWidth(1,250);
table1->setColumnWidth(2,100);
table1->setColumnWidth(3,100);
table1->setColumnWidth(4,180);
for(i=1;i<=table1->rowCount();i++) //遍历表格
{
  checkBox=new QTableWidgetItem(); //建立选择框
    checkBox->setCheckState(Qt::Unchecked); //设置选择框
//状态
  table1->setItem(i-1,0,checkBox); //加入选择框
}
table1->setGeometry(QRect(20,100,700,360)); //设置表格位
//置
table1->setItem(i-1,1,new QTableWidgetItem(s)); //设置单
//元格文字
s=curpath+"/"+table1->item(i-1,1)->text(); //获取单元格文
//字
```

表格控件用来显示表格形式的数据，是一种功能比较强的控件。表格由若干行，若干列组成。每个单元格中可以是文字，也可以是 Qt 中的各种控件。

3 TAR 文件简介

TAR 文件是 Linux 系统下一种常见的归档文件。它可以将多个目录和文件集中到一个文件中。TAR 文件由块构成，每个块有 512 字节。每个文件，首先是 TAR 文件头，然后是文件内容，文件内容也以 512 字节为一块，不足 512 字节用 0 进行填充。全部文件存储结束后，加入一个全 0 的块。

TAR 头结构如下：

```
struct tar_header //TAR 文件头
{
  char name[100]; //名字
  char mode[8]; //模式
  char uid[8]; //用户号
  char gid[8]; //组号
  char size[12]; //大小
  char mtime[12]; //修改时间
  char chksum[8]; //校验和
  char typeflag; //类别 5-目录 0-文件
  char linkname[100]; //链接名,用 0 填充
  char magic[6]; //标识,'ustar '
  char version[2]; //版本,用 0 填充
  char uname[32]; //用户名
  char gname[32]; //组名
  char devmajor[8]; //主设备号,用 0 填充
  char devminor[8]; //次设备号,用 0 填充
  char prefix[155]; //前缀,用 0 填充
  char padding[12]; //填充项,用 0 填充
};
```

其中，name 是文件或目录名。mode 是文件存储模式，8 进制存储。uid 和 gid 是用户号和组号，8 进制存储。size 是文件大小，8 进制存储。mtime 是文件修改时间，8 进制存储。chksum 用于校验 TAR 文件，它的计算方法是 chksum 用 8 个空格填充，512 个字节的 ASCII 码之和。

4 查找文件和目录

使用代码如下：

```
//取文件和目录,不包括子目录中文件
void tarman::getfiles(const QString &path,QStringList *list1)
{
  QDir dir(path);
  int nFiles=0,i=0;
  bool bisDir;
  QFileInfoList list;
  QFileInfo fileInfo;
  if(! dir.exists()) return;
  dir.setFilter(QDir::Dirs|QDir::Files);
  dir.setSorting(QDir::DirsFirst);
  list=dir.entryInfoList();
  do
  {
    fileInfo=list.at(i);
    if(fileInfo.fileName()=="."||fileInfo.fileName()="..")
    {
      i++;
      continue;
    }
    bisDir=fileInfo.isDir();
    if(bisDir)
    {
      nFiles++;
      list1->append("5"+fileInfo.path()+"/"+fileInfo.fileName());
    }
    else
    {
      nFiles++;
      list1->append("0"+fileInfo.path()+"/"+fileInfo.fileName());
    }
    i++;
  }while(i<list.size());
}
```

如果要包括子目录中文件，只要增加一个递归调用即可：

```
if(bisDir)
{
  nFiles++;
  list1->append("5"+fileInfo.path()+"/"+fileInfo.fileName());
```

```
  getallfiles(fileInfo.filePath(),list1);  //递归调用
}
```

5 获取用户和组

使用如下代码获取用户和组：

```
//取用户和组
stream=popen("id","r");
r=fread(buf,1,1000,stream);
pclose(stream);
if(strstr(buf,"uid=")!  =NULL)
{
 for(i=1;i<=1000;i++)
 {
  if(buf[i-1]=='u'&&buf[i]=='i'&&
     buf[i+1]=='d'&&buf[i+2]=='=')
  {
    strcpy(buf1,buf+i+3);
    break;
  }
 }
 for(i=1;i<=8;i++)
 {
  if(buf1[i-1]=='(')
  {
    strcpy(buf2,buf1+i);
    for(j=1;j<=32;j++)
    {
     if(buf2[j-1]==')')
     {
       buf2[j-1]='\0';
       strcpy(uname,buf2);
       uname[strlen(uname)]='\0';
       break;
     }
    }
    buf1[i-1]='\0';
    strcpy(uid,buf1);
    break;
  }
 }
}
if(strstr(buf,"gid=")!  =NULL)
{
 for(i=1;i<=1000;i++)
 {
  if(buf[i-1]=='g'&&buf[i]=='i'&&
     buf[i+1]=='d'&&buf[i+2]=='=')
  {
    strcpy(buf1,buf+i+3);
    break;
  }
```

```
 }
 for(i=1;i<=8;i++)
 {
  if(buf1[i-1]=='(')
  {
    strcpy(buf2,buf1+i);
    for(j=1;j<=32;j++)
    {
     if(buf2[j-1]==')')
     {
       buf2[j-1]='\0';
       strcpy(gname,buf2);
       gname[strlen(gname)]='\0';
       break;
     }
    }
    buf1[i-1]='\0';
    strcpy(gid,buf1);
    break;
  }
 }
}
```

6 关键代码

向 TAR 文件中加入文件和目录代码如下：

```
void tarman::exec_button1_1()  //执行加入按钮
{
 QString str,s,s4;
 int i,j,l,r,r1;
 QStringList *list;
 FILE *fp,*fp1;
 char s2[250],s3[100],buf[512];
 unsigned int k;
 struct tar_header tarbuf;
 unsigned long sum;
 struct stat buffer;

 sprintf(s2,"%s",tarfilename.toLocal8Bit().data());
 fp=fopen(s2,"wb");
 //加入根目录
 strcpy(tarbuf.name,"./");
 for(k=1;k<=100-strlen(tarbuf.name);k++)
   tarbuf.name[k-1+strlen(tarbuf.name)]='\0';
 sprintf(tarbuf.mode,"%07o",509);
 sprintf(tarbuf.uid,"%07o",atoi(uid));
 sprintf(tarbuf.gid,"%07o",atoi(gid));
 sprintf(tarbuf.size,"%011o",0);
 sprintf(tarbuf.mtime,"%011o",0);
 memset(tarbuf.chksum,32,8);
 tarbuf.typeflag='5';
 memset(tarbuf.linkname,0,100);
```

```
strcpy(tarbuf.magic,"ustar ");
tarbuf.version[0]=' ';
tarbuf.version[1]='\0';
strcpy(tarbuf.uname,uname);
for(k=1;k<=100-strlen(tarbuf.uname);k++)
  tarbuf.uname[k-1+strlen(tarbuf.uname)]='\0';
strcpy(tarbuf.gname,gname);
for(k=1;k<=100-strlen(tarbuf.gname);k++)
  tarbuf.gname[k-1+strlen(tarbuf.gname)]='\0';
memset(tarbuf.devmajor,0,8);
memset(tarbuf.devminor,0,8);
memset(tarbuf.prefix,0,155);
memset(tarbuf.padding,0,12);
sum=0;
memcpy(buf,&tarbuf,512);
for(j=1;j<=512;j++) sum=sum+buf[j-1];
sprintf(tarbuf.chksum,"%06lo",sum);
r=fwrite(&tarbuf,1,512,fp);
for(i=1;i<=table1->rowCount();i++)
{
 if(table1->item(i-1,0))
 {
  if(table1->item(i-1,0)->checkState()==Qt::Checked)
  {
   if(table1->item(i-1,1))
   {
    if(table1->item(i-1,2))
    {
     //加入当前目录下文件
if(table1->item(i-1,2)->text()==str.fromLocal8Bit("文件"))
      {
       s=curpath+"/"+table1->item(i-1,1)->text();
       sprintf(s2,"%s",s.toLocal8Bit().data());
       sprintf(s3,"%s",
    table1->item(i-1,1)->text().toLocal8Bit().data());
       r=stat(s2, &buffer);
       strcpy(tarbuf.name,"./");
       strcat(tarbuf.name,s3);
       for(k=1;k<=100-strlen(tarbuf.name);k++)
        tarbuf.name[k-1+strlen(tarbuf.name)]='\0';
       sprintf(tarbuf.mode,"%07o",buffer.st_mode);
       sprintf(tarbuf.uid,"%07o",atoi(uid));
       sprintf(tarbuf.gid,"%07o",atoi(gid));
       sprintf(tarbuf.size,"%011o",
            (unsigned int)buffer.st_size);
       sprintf(tarbuf.mtime,"%011o",
            (unsigned int)buffer.st_mtime);
       memset(tarbuf.chksum,32,8);
       tarbuf.typeflag='0';
       memset(tarbuf.linkname,0,100);
       strcpy(tarbuf.magic,"ustar ");
       memset(tarbuf.version,0,2);
       strcpy(tarbuf.uname,uname);
       for(k=1;k<=100-strlen(tarbuf.uname);k++)
        tarbuf.uname[k-1+strlen(tarbuf.uname)]='\0';
       strcpy(tarbuf.gname,gname);
       for(k=1;k<=100-strlen(tarbuf.gname);k++)
        tarbuf.gname[k-1+strlen(tarbuf.gname)]='\0';
       memset(tarbuf.devmajor,0,8);
       memset(tarbuf.devminor,0,8);
       memset(tarbuf.prefix,0,155);
       memset(tarbuf.padding,0,12);
       sum=0;
       memcpy(buf,&tarbuf,512);
       for(k=1;k<=512;k++) sum=sum+buf[k-1];
       sprintf(tarbuf.chksum,"%06lo",sum);
       r=fwrite(&tarbuf,1,512,fp);
       fp1=fopen(s2,"rb");
       if(fp1!  =NULL)
       {
        while(1)
        {
         r=fread(buf,1,512,fp1);
         if(r>0)
         {
          for(l=1;l<=512-r;l++) buf[r+l-1]='\0';
          r1=fwrite(buf,1,512,fp);
         }
          if(r!  =512) break;
        }
        fclose(fp1);
       }
      }
      //加入当前目录下目录
if(table1->item(i-1,2)->text()==str.fromLocal8Bit("目录"))
      {
       s=curpath+"/"+table1->item(i-1,1)->text();
       s4="./"+table1->item(i-1,1)->text();
       sprintf(s2,"%s",s4.toLocal8Bit().data());
       strcpy(tarbuf.name,s2);
       for(k=1;k<=100-strlen(tarbuf.name);k++)
        tarbuf.name[k-1+strlen(tarbuf.name)]='\0';
       sprintf(tarbuf.mode,"%07o",509);
       sprintf(tarbuf.uid,"%07o",atoi(uid));
       sprintf(tarbuf.gid,"%07o",atoi(gid));
       sprintf(tarbuf.size,"%011o",0);
       sprintf(tarbuf.mtime,"%011o",0);
       memset(tarbuf.chksum,32,8);
       tarbuf.typeflag='5';
       memset(tarbuf.linkname,0,100);
       strcpy(tarbuf.magic,"ustar ");
       memset(tarbuf.version,0,2);
       strcpy(tarbuf.uname,uname);
       for(k=1;k<=100-strlen(tarbuf.uname);k++)
```

```
  tarbuf.uname[k-1+strlen(tarbuf.uname)]='\0';
strcpy(tarbuf.gname,gname);
for(k=1;k<=100-strlen(tarbuf.gname);k++)
  tarbuf.gname[k-1+strlen(tarbuf.gname)]='\0';
memset(tarbuf.devmajor,0,8);
memset(tarbuf.devminor,0,8);
memset(tarbuf.prefix,0,155);
memset(tarbuf.padding,0,12);
sum=0;
memcpy(buf,&tarbuf,512);
for(k=1;k<=512;k++) sum=sum+buf[k-1];
sprintf(tarbuf.chksum,"%06lo",sum);
r=fwrite(&tarbuf,1,512,fp);
list=new QStringList;
getallfiles(s,list);
for(j=1;j<=list->size();j++)
{
 s4=list->value(j-1);
 if(s4.left(1)=="5") //加入子目录下目录
 {
   s4.remove(0,1);
   s4="./"+s4;
   s4.replace(curpath+"/","");
   sprintf(s2,"%s",s4.toLocal8Bit().data());
   strcpy(tarbuf.name,s2);
for(k=1;k<=100-strlen(tarbuf.name);k++)
  tarbuf.name[k-1+strlen(tarbuf.name)]='\0';
sprintf(tarbuf.mode,"%07o",509);
sprintf(tarbuf.uid,"%07o",atoi(uid));
sprintf(tarbuf.gid,"%07o",atoi(gid));
sprintf(tarbuf.size,"%011o",0);
sprintf(tarbuf.mtime,"%011o",0);
memset(tarbuf.chksum,32,8);
tarbuf.typeflag='5';
memset(tarbuf.linkname,0,100);
strcpy(tarbuf.magic,"ustar ");
memset(tarbuf.version,0,2);
strcpy(tarbuf.uname,uname);
for(k=1;k<=100-strlen(tarbuf.uname);k++)
  tarbuf.uname[k-1+strlen(tarbuf.uname)]='\0';
strcpy(tarbuf.gname,gname);
for(k=1;k<=100-strlen(tarbuf.gname);k++)
  tarbuf.gname[k-1+strlen(tarbuf.gname)]='\0';
memset(tarbuf.devmajor,0,8);
memset(tarbuf.devminor,0,8);
memset(tarbuf.prefix,0,155);
memset(tarbuf.padding,0,12);
sum=0;
memcpy(buf,&tarbuf,512);
for(k=1;k<=512;k++) sum=sum+buf[k-1];
sprintf(tarbuf.chksum,"%06lo",sum);
r=fwrite(&tarbuf,1,512,fp);
 }
 if(s4.left(1)=="0") //加入子目录下文件
 {
   s4.remove(0,1);
   sprintf(s2,"%s",s4.toLocal8Bit().data());
   r=stat(s2, &buffer);
   s4="./"+s4.replace(curpath+"/","");
   sprintf(s3,"%s",s4.toLocal8Bit().data());
   strcpy(tarbuf.name,s3);
for(k=1;k<=100-strlen(tarbuf.name);k++)
  tarbuf.name[k-1+strlen(tarbuf.name)]='\0';
sprintf(tarbuf.mode,"%07o",buffer.st_mode);
sprintf(tarbuf.uid,"%07o",atoi(uid));
sprintf(tarbuf.gid,"%07o",atoi(gid));
sprintf(tarbuf.size,"%011o",
      (unsigned int)buffer.st_size);
sprintf(tarbuf.mtime,"%011o",
      (unsigned int)buffer.st_mtime);
memset(tarbuf.chksum,32,8);
tarbuf.typeflag='0';
memset(tarbuf.linkname,0,100);
strcpy(tarbuf.magic,"ustar ");
memset(tarbuf.version,0,2);
strcpy(tarbuf.uname,uname);
for(k=1;k<=100-strlen(tarbuf.uname);k++)
  tarbuf.uname[k-1+strlen(tarbuf.uname)]='\0';
strcpy(tarbuf.gname,gname);
for(k=1;k<=100-strlen(tarbuf.gname);k++)
  tarbuf.gname[k-1+strlen(tarbuf.gname)]='\0';
memset(tarbuf.devmajor,0,8);
memset(tarbuf.devminor,0,8);
memset(tarbuf.prefix,0,155);
memset(tarbuf.padding,0,12);
sum=0;
memcpy(buf,&tarbuf,512);
for(k=1;k<=512;k++) sum=sum+buf[k-1];
sprintf(tarbuf.chksum,"%06lo",sum);
r=fwrite(&tarbuf,1,512,fp);
fp1=fopen(s2,"rb");
if(fp1!=NULL)
{
while(1)
{
  r=fread(buf,1,512,fp1);
  if(r>0)
  {
    for(l=1;l<=512-r;l++)
    buf[r+l-1]='\0';
    r1=fwrite(buf,1,512,fp);
  }
  if(r!=512) break;
}
```

```
            fclose(fp1);
          }
        }
      }
    }
  }
 }
}
 }
 memset(buf,0,512);
 r=fwrite(buf,1,512,fp);
 fclose(fp);
 QMessageBox::information(NULL,str.fromLocal8Bit("消息"),
                  str.fromLocal8Bit("加入成功"),
                  QMessageBox::Ok,QMessageBox::Ok);
 dlg1->close();
}
```

从 TAR 文件中解出文件和目录代码如下：

```
void tarman::exec_button2_1()  //执行解出按钮
{
 QString str;
 FILE *fp,*fp1=NULL;
 char s[250],s1[100],s2[250],buf[512],fname[100];
 int i,j,r,r1,flag;
 unsigned long k;
 struct tar_header tarbuf;
 unsigned long filesize=0,blockcount=0,filetime=0,filemode=
0;
 struct utimbuf timebuf;
 sprintf(s,"%s",tarfilename.toLocal8Bit().data());
 fp=fopen(s,"rb");
 if(fp! =NULL)
 {
  for(i=1;i<=table2->rowCount();i++)
  {
   r=fread(&tarbuf,1,512,fp);
   strcpy(fname,tarbuf.name);
   memcpy(buf,&tarbuf,512);
   flag=1;
   for(j=1;j<=512;j++)
   {
 if(buf[j-1]! =0)
 {
  flag=0;
  break;
 }
}
if(flag==1) break;
if(tarbuf.typeflag=='5')
{
  filesize=0;
  filetime=0;
  filemode=0;
  blockcount=0;
}
if(tarbuf.typeflag=='0')
{
  filesize=strtol(tarbuf.size,NULL,8);
  filetime=strtol(tarbuf.mtime,NULL,8);
  filemode=strtol(tarbuf.mode,NULL,8);
  blockcount=filesize/512+1;
  if(filesize%512==0) blockcount--;
}
//解出选中文件
if(table2->item(i-1,0)->checkState()==Qt::Checked)
{
 if(table2->item(i-1,1))
 {
  sprintf(s,"%s",tarfilename.toLocal8Bit().data());
  for(j=strlen(s);j>=0;j--)
  {
   if(s[j-1]=='/')
   {
    s[j-1]='\0';
    break;
   }
  }
      sprintf(s1,"/%s",
            table2->item(i-1,1)->text().toLocal8Bit().data());
 if(s1[0]=='/'&&s1[1]=='.'&&s1[2]=='/') strcpy(s1,s1+2);
      if(strcmp(s1,"/./")! =0) strcat(s,s1);
      if(s[strlen(s)-1]=='/') s[strlen(s)-1]='\0';
      if(table2->item(i-1,2))
      {
       //解出目录
 if(table2->item(i-1,2)->text()==str.fromLocal8Bit("目录"))
       if(opendir(s)==NULL) mkdir(s,0775);
       //解出文件
 if(table2->item(i-1,2)->text()==str.fromLocal8Bit("文件"))
       {
        sprintf(s2,"%s",tarfilename.toLocal8Bit().data());
        for(j=strlen(s2);j>=0;j--)
        {
         if(s2[j-1]=='/')
         {
          s2[j]='\0';
          break;
         }
        }
        if(fname[0]=='.'&&fname[1]=='/')
         strcat(s2,fname+2);
        else
```

```
            strcat(s2,fname);
            if(strcmp(s2,s)==0) fp1=fopen(s2,"wb");
          }
        }
      }
    }
    for(k=1;k<=blockcount;k++)
    {
      r=fread(buf,1,512,fp);
      if(fp1! =NULL)
      {
        if(r>0)
        {
          if(k! =blockcount)
            r1=fwrite(buf,1,512,fp1);
          else
            r1=fwrite(buf,1,filesize%512,fp1);
        }
      }
      if(r! =512) break;
    }
    if(fp1! =NULL)
    {
      fclose(fp1);
      fp1=NULL;
      timebuf.actime=time(NULL);
      timebuf.modtime=filetime;
      utime(s2,&timebuf);
      chmod(s2,filemode);
    }
  }
  fclose(fp);
  QMessageBox::information(NULL,str.fromLocal8Bit("消息"),
                           str.fromLocal8Bit("解出成功"),
                           QMessageBox::Ok,QMessageBox::Ok);
  dlg2->close();
 }
}
```

7 结语

通过综合使用前面所述的技巧，实现了一个简单实用型的Linux图形界面下的应用程序。程序可以实现TAR文件和目录加入和解出操作。本例程对于在Linux图形模式下开发类似的应用程序有参考价值。

（收稿日期：2012-09-20）

全新的用户体验，瑞星杀毒软件V16成功上线！

著名的杀毒软件厂商瑞星不久前发布了第十六代瑞星杀毒软件。该版本一经推出就获得了业界的一致好评，彻底改变了界面操作困难、不友好、响应慢等令用户头痛的问题。在瑞星官网可以发现，这次版本的升级中他们组建了全新的开发团队，由行业资深专家和专业领域的博士、硕士人员组成。是什么样的专家令瑞星界面的用户体验一举改变？采用了什么样的开发方式在短短的几个月里令瑞星杀毒软件有了这么大的提升？瑞星公司的高层给出了答案：

瑞星V16版本是一个完全着眼于个人用户市场的产品，所以在用户体验上必须做到行业的No.1。而从以往版本的用户反馈来看，有70%以上都是界面相关的问题。瑞星杀毒软件在中国杀毒行业里面是领先的牌子之一，它有大量的用户基础，用户对它热爱之余也提出了很多改进的建议，比如界面看上去比较死板不够专业等。甚至有些用户因为这些问题而导致放弃使用，这对瑞星公司来说则是非常严重的事情。如何快速地解决这些问题就成为摆在瑞星公司高层面前最主要的问题了。而在2年前，瑞星公司的技术人员曾经尝试通过优化界面代码来提升界面，但最后效果不太理想。瑞星高层对此十分重视，经过审慎的研究、分析，结论是要解决这个问题必须从根上系统地来解决，而不能像之前只是解决表面的一些东西。高层首先想到的是对内求提升，花费重金组建了专业的用户体验团队，包括用户研究组、交互设计策划组、视觉设计创新组和界面开发组。然后想到的是对外请外脑，采购国内优秀的界面开发工具和专业的UI设计团队的整体能力培训服务。这样一内一外的组合正好实现了优势互补，为此次界面的改造打下了坚实的基础。经过整整三个月的奋力拼搏，瑞星杀毒软件V16 Beta版于2012年10月20日发布。通过对用户的反馈进行整理后发现，此次版本的升级让用户感觉极有进步的是用户体验的全面提升。也正是通过这次版本的历练，瑞星高层深深感受到真正的专业不只是背后的实力，更应从细枝末节中体现出来。

这里大家可能都会有个疑问：瑞星公司此次的外脑是谁？又是怎么样如此快速地帮助他们组建UI设计和开发团队？对此瑞星高层却是讳莫如深。

经过多方的求证，最后终于找出，此次为瑞星公司提供界面服务的原来是上海勇进软件有限公司（UIPower.com）。通过该公司的网站，发现了该公司还曾为多家大型软件公司，如华为、中兴、步步高、盛大网络等提供过类似服务，为其在团体建设和改善用户体验方面都做出不少贡献。

自动关机助手软件设计与实现

覃盈保　韩 俊

摘　要：通过对自动关机需求的分析，利用 C# 开发了能实现自动运行、自动关机的软件，实现了软件的自动关机功能。

关键词：关机软件；托盘图标；时钟控件；注册表；C# 语言；外接程序；输入验证

随着信息化的发展，电脑开机的时间和次数也越来越多，这对节约电能是个巨大的挑战。由于个人的有意和无意的行为，电脑长时间开机而无利用，造成了电量的巨大浪费。若电脑能在开机时自动运行一个关机程序，在指定的关机时间能自动关机，这部分浪费掉的电能就可以节约下来，对计算机也起到保护作用。下文在 Visual Studio 2010 中利用 C#，实现了电脑的自动关机功能。

1　自动关机软件功能流程

软件的整体功能流程如图 1 所示。

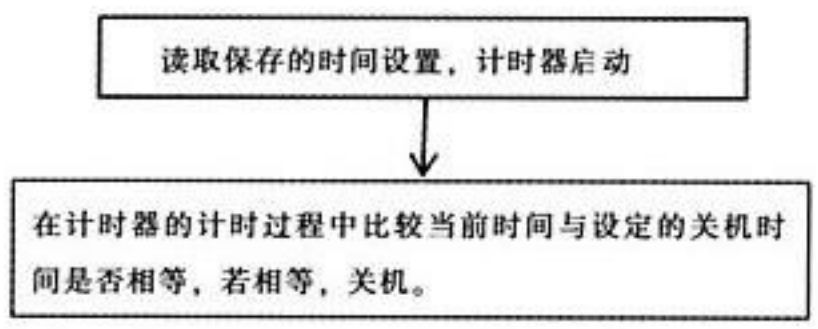


图 1　软件的整体功能流程图

为了使软件在运行过程中不影响电脑桌面，程序不在任务栏显示，而显示在任务栏的通知区域中，并且设置了快捷操作。

2　软件设计

2.1　软件界面

软件界面及运行中的托盘图标如图 2 所示。

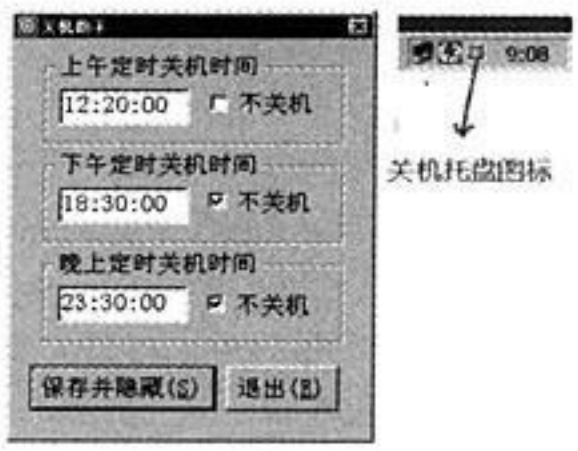


图 2　软件界面

为了方便设定时间，设计了 3 个输入文本框，添加的隐藏控件有 NotifyIcon 组件，右键菜单 ContextMenuStrip，计时器控件 Timer，ErrorProvider 控件（提供用于指示窗体上的控件具有关联错误的用户界面）。

NotifyIcon 组件的属性设置：

```
notifyIcon1.BalloonTipIcon = ToolTipIcon.Info;
notifyIcon1.ContextMenuStrip = contextMenuStrip1;//关联
//的右键菜单
        notifyIcon1.Icon = ((System.Drawing.Icon)(resources.
GetObject("notifyIcon1.Icon")));
notifyIcon1.Text = "自动关机助手";
notifyIcon1.Visible = true;
                    notifyIcon1.MouseDoubleClick += new
MouseEventHandler(this.notifyIcon1_MouseDoubleClick);
```

右键菜单 contextMenuStrip1，功能主要有：弹出程序：主窗口（TSMMainWindow）、退出程序：退出（TSMExit）。

计时器控件 Timer 的属性设置：

```
timer1.Interval = 1000;//间隔 1000 毫秒比较一次时间
 timer1.Tick  +=  new  System.EventHandler  (this.
timer1_Tick);
```

2.2　需要特别导入的命名空间

```
using System.IO;
using Microsoft.Win32;
```

2.3　保存和读取当前设置的时间

其函数是 SaveTime () 和 LoadTime ()，保存设置的时间到当前目录下的 TimeSet.txt 文件中，程序加载和弹出主窗口时进行读取。

2.4　初始设置

窗体加载时，即在窗体的 Load 函数中，要读取设置时间、启动计时器、设置注册表的启动项，其代码如下：

```
private void MainForm_Load(object sender, EventArgs e)
 {
    LoadTime();//读取设置的时间
    timer1.Enabled = true;// 设置计时器可以运行
    timer1.Start();      //启动计时器
        //检查注册表是否已设置了相关的启动项
        RegistryKey  RKey  =  R egistry.LocalMachine.
```

```
OpenSubKey("SOFTWARE\\Microsoft\\" +
            "Windows\\CurrentVersion\\Run", true);
          if(RKey==null)
              RKey = Registry.LocalMachine.OpenSubKey ("
SOFTWARE\\Microsoft\\" +
            "Windows\\CurrentVersion\\Run");
          RKey.SetValue ("自动关机助手", Application.
StartupPath + "\\AutoShutdown.exe");
            EnableChecked();//由当前要比较的关机时间节点
    }
```

2.5 调用外部程序

电脑关机时用到的外接程序是 shutdown.exe，在命令提示符窗口下运行，可以关闭计算机，如“shutdown -f -s -t 0”表示对计算机进行强制关机，延时 0 秒。

要运行该外接程序，要启动一个线程，其代码如下：

```
    //关闭计算机
  private void shutDown()
   {
       System.Diagnostics.Process myProces = new
System.Diagnostics.Process();
      myProces.StartInfo.FileName = "cmd.exe";
      myProces.StartInfo.RedirectStandardInput = true;
      myProces.StartInfo.RedirectStandardOutput = true;
      myProces.StartInfo.RedirectStandardError = true;
      myProces.StartInfo.UseShellExecute = false;
      myProces.StartInfo.CreateNoWindow = true;
      myProces.Start();
myProces.StandardInput.WriteLine("shutdown -f -s -t 0");
    }
```

2.6 时间确认和提醒

要进行两个时间节点的比较，一个是指定关机前一分钟的时间比较，另一个是指定关机时间的比较。比较时间在计时器的 Tick 方法中进行，定义函数 int compareTime ()，返回值为 0 定义为关机时间到了，执行关机子程序 shutDown ()。每个指定的时间间隔（1000 毫米，即 1 秒），分别对当前要比较的时间节点设置值进行比较。为了提前预警，设置了提前一分钟预警窗口，如图 3 所示。如判别上午关机时间的代码如下：

```
        string errorMsg;
      DateTime nowTime = DateTime.Now;
              TimeSpan nowTimeSpan = new TimeSpan
(nowTime.Hour, nowTime.Minute,
                    nowTime.Second);

      if (! chBForenoon.Checked)
      {
                    if  (ValidTextBox (tBForenoonTime,
tBForenoonTime.Text, out errorMsg))
          {
  targetTime = getSpanTime(tBForenoonTime.Text.Trim());
              if ((targetTime.TotalMinutes - nowTimeSpan.
TotalMinutes) == 1)
              {//提前一分钟预警
  AlertForm ff = new AlertForm(targetTime.TotalSeconds);
      DialogResult result = ff.ShowDialog();//显示警告窗口
      if (result==DialogResult.Yes)//取消本次关机设定
              {
                  chBForenoon.Checked =true;
                  ff.Close();
                  return 1;
              }
          }
 if (nowTimeSpan.CompareTo(targetTime) == 0)
          {
return 0;//关机时间已到,返回执行关机子程序。
          }
          if (nowTimeSpan.CompareTo(targetTime) > 0)
          {//已过了关机时间,设定不比较本次关机时间
              chBForenoon.Checked = true;
          }
      }
  }
```

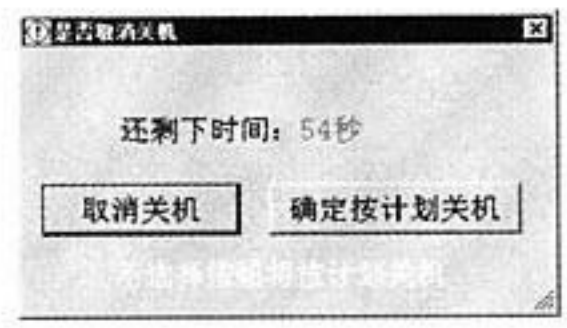


图 3 一分钟提前预警窗口

2.7 输入的时间格式验证

要比较的时间的小时必须是在 0 到 23 之间的整数，分钟和秒钟是在 0 到 59 之间的整数，TextBox 的验证事件有两个，一个是验证中的即 Validating，另一个是验证后的即 Validated，这两个事件发生在 TextBox 文本框失去焦点后才发生，要测试即时输入的验证，添加 TextChanged 事件。下面列出这 3 个事件的代码，以上午输入文本框为例：

```
    //上午时间输入改变时
    private void tBForenoonTime_TextChanged (object
sender, EventArgs e)
    {
        string errorMsg;
        if (! ValidTextBox(tBForenoonTime,tBForenoonTime.
Text, out errorMsg))
        {
    this.errorProvider1.SetError(tBForenoonTime, errorMsg);
        }
    }
    //上午时间输入进行验证中
    private void tBForenoonTime_Validating (object sender,
CancelEventArgs e)
```

```
    {
        string errorMsg;
        if (! ValidTextBox(tBForenoonTime, tBForenoonTime.
Text, out errorMsg))
        {
            e.Cancel = true;//取消本次输入
    this.errorProvider1.SetError(tBForenoonTime, errorMsg);
        }
    }
    //上午时间输入验证后,设定提示信息为空
     private void tBForenoonTime_Validated (object sender,
EventArgs e)
    {
        this.errorProvider1.SetError(tBForenoonTime, "");
    }
```

函数 bool ValidTextBox (TextBox sender,string strTextBox, out string errorMessage) 的作用是验证文本框的输入是否符合指定的时间格式，以测试小时部分的时间为例，其代码如下：

```
     public bool ValidTextBox (TextBox sender,string
strTextBox, out string errorMessage)
    {
string[] strArray = strTextBox.Split(new char[] { ':', ' ' }, 3);
        int result = 0;
        if (strTextBox.Length == 0)
        {
errorMessage = "需要设置正确的时间格式,如 12:12:12.";
            return false;
        }
        //验证小时时间
        try
        {
            result = Convert.ToInt32(strArray[0]);
        }
        catch(FormatException)
        {
            errorMessage = "小时时间需要输入整数字.";
            sender.Select(0, strArray[0].Length);
            return false;
        }
        //控制输入的小时部分符合上午、中午、或晚上
        if (sender.Name == "tBForenoonTime")
        {
            if (result > 12||result<0)
            {
errorMessage = "上午小时时间数字在 00 到 12 之间.";
                sender.Select(0, strArray[0].Length);
                return false;
            }
        }
        if (sender.Name == "tBAfternoonTime")
        {
            if (result > 18 || result < 13)
            {
errorMessage = "中午小时时间数字在 13 到 18 之间.";
                sender.Select(0, strArray[0].Length);
                return false;
            }
        }
        if (sender.Name == "tBNightTime")
        {
            if (result > 23 || result < 19)
            {
errorMessage = "晚上小时时间数字在 19 到 23 之间.";
                sender.Select(0, strArray[0].Length);
                return false;
            }
        }
        ...
EnableChecked();//判断当前的设置的时间是否是目标关机点
        errorMessage = "";//没有错误,返回空信息
        return true;
    }
```

2.8 读取输入间隔时间

从文本框的时间字符串中取得时间间隔的函数代码如下：

```
    private TimeSpan getSpanTime(string str)
    {
        TimeSpan timeSpan=new TimeSpan(0,0,0);
        if (str.Length >= 6)
        {
            string[] strArray = str.Split(new char[] { ':', ' ' }, 3);
            try
            {
                timeSpan = new TimeSpan (Convert.ToInt32
(strArray[0]), Convert.ToInt32(strArray[1]),
                                Convert.ToInt32(strArray[2]));
            }
        }
        return timeSpan;
    }
```

2.9 调用主界面

当双击托盘图标或右击托盘图标弹出右键菜单选择“主窗口”时，显示软件界面，其代码如下：

```
  if (this.WindowState == FormWindowState.Minimized)
            this.WindowState = FormWindowState.Normal;
        this.Activate();
        this.Show();
        this.Visible = true;
        this.TopMost = true;
```

2.10 自动后台加载

要使电脑开机加载本软件而不显示界面，只要在主窗口的 shown 方法中增加语句：this.Hide () 即可。

(下转第 93 页)

DDE 在 LonWorks 网络监控程序中的应用

李志远

摘　要：介绍了 LonWorks 网络监控程序的实现技术 DDE 及其用到的 LNS DDE Server 的工作原理和实现方式进行了介绍，并给出了监控程序实现的的技术途径和步骤。

关键词：DDE 技术； LonWorks 网络； DDEML 库； 监控程序

1　引言

DDE（Dynamic Data Exchange）技术是微软提供的在 Windows 环境下基于消息的进程间的通信技术协议。以强大的客户/服务器（Client/Server）体系结构为基础，也是支持在 Windows 环境下多种编程平台（VC、VB、Excel、PB 等）程序的集成和相互访问的重要技术手段。LonWorks 网络是由美国 Echelon 公司提供的一个开放的控制网络平台技术，是一个开放性和互操作性很强的、无专利权的低层通信网络--局部操作网络，简称 LON，是全世界最为普遍的用来连接日常设备的标准。

进行 LonWorks 网络监控程序的开发实现手段主要有以下几种：

（1）OpenLDV SDK：该开发包完全免费，有着完整的技术资料，无技术支持，适用于研究型的课题，对工程应用来说开发难度和开发周期不易控制。

（2）LNS Turbo Edition Application Developer's Kit：该开发包价格昂贵，基于 OLE 技术，有技术支持和成熟应用案例，对工程应用来说开发简单、高效。

（3）LNS DDE Server：该开发包价格适中，基于 DDE 技术，有技术支持和成熟的应用案例，开发周期短，适用于网络节点不是特别多的场合。

本应用 LonWorks 网络节点数量最多为 250 个，无实时性的要求，考虑到系统的经济型采用 LNS DDE Server 进行监控程序的开发应用。

从发展趋势来看，基于 OLE 的数据交换是最好的，它特别符合当今软件领域的客户-服务器机制 (Client-Server)。基于传统的 DDE 数据交换因其应用简单，目前仍然有着广泛的应用空间。

2　LNS DDE Server 和 Client

LNS DDE Server 是 Echelon 公司提供的基于 DDE 技术开发的服务器，可与任何 Windows 环境下编程平台开发的基于 DDE 技术的客户端进行通信。通过建立 DDE 协议的连接，可实现对 LonWorks 网络设备进行相互操作，具体包括：

（1）读、监视和修改任何网络变量的值。

（2）监视和改变配置属性。

（3）接收和发送应用程序报文。

（4）测试 (Test)、启用 (Enable)、禁用 (Disable) 以及强制 (Override) LonMark 对象。

（5）测试、闪烁 (Wink) 以及控制设备。

在进行 DDE 客户端访问 LNS DDE Server 前首先要利用 LonMaker for Windows 集成网络管理工具对 LonWorks 网络中的设备进行应用程序的下载、设备安装、网络变量连接、报文标记、基本的网络诊断和控制等，LonMaker 工具是基于 LNS 网络操作系统，其配置网络结构采用简单易用的 Visio 用户界面，具体如图 1 所示。一旦 LonWorks 网络由 LonMaker for Windows 配置正确完毕后投入使用，LNS DDE Server 才可以自动访问由 LonMaker 工具自动创建的 LNS 数据库。

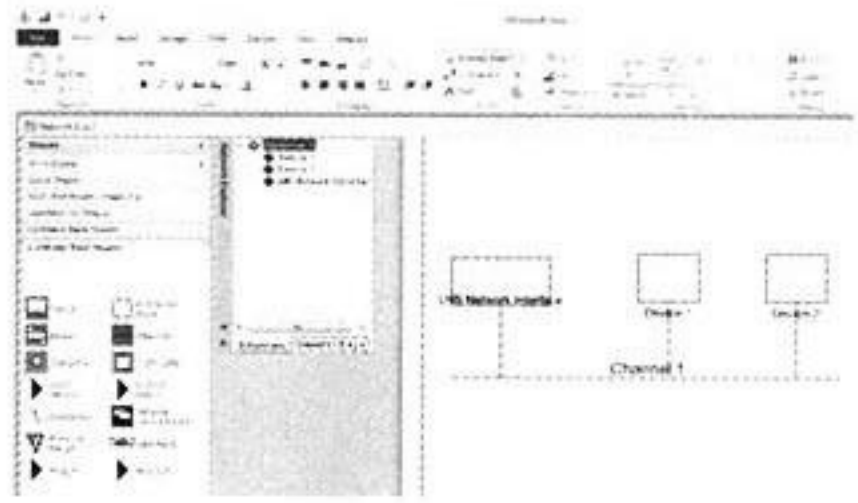


图 1　LonMaker 网络配置 Visio 用户界面

当客户端（Client）与服务器（Server）通过 DDE 进行数据交换时成为对话，客户端负责初始化与服务器的会话及控制会话流，服务器负责响应客户端的请求，客户端就可以进行数据的发送和接收，完成数据的交互。进行 DDE 通信的客户端应用程序采用程序名（Application）、主题名（Topic）和项目名（Item），应用程序名和主题名确定一次 DDE 会话，项目名确定会话时需要交换的共享数据。

DDE 的客户端和服务器之间的数据传递为隐式报文格式，其缺点是对于文件、图像数据的传递靠网络变量的数据传递来

实现，要保证传输文件的正确性，需对传输的数据个数、数据的校正及数据的解析实现来说就麻烦一些。

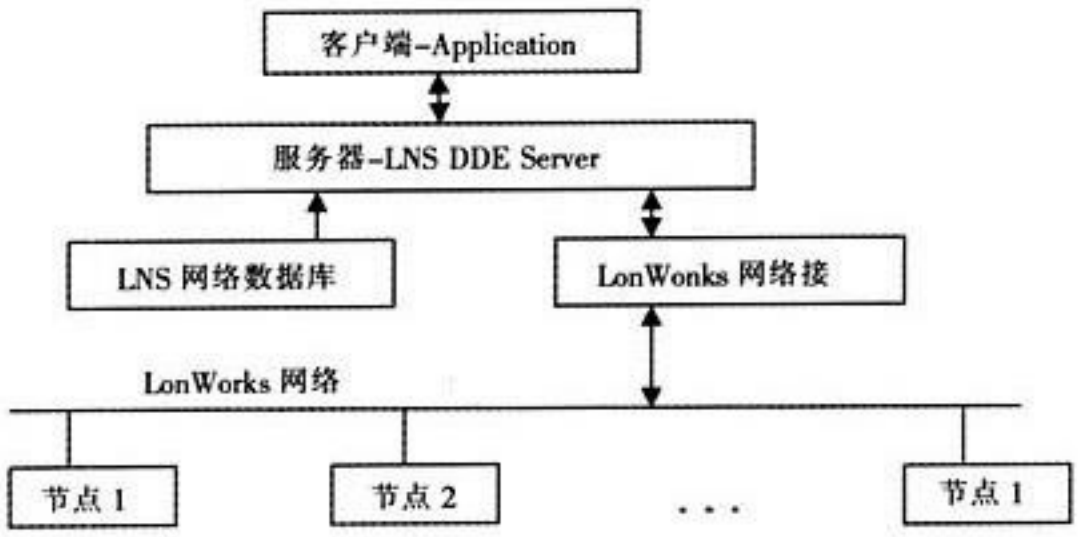


图 2 LNS DDE Server 工作原理

DDE 客户端向服务器发出的请求内容包括：

Initiate：请求建立一条 DDE 会话通道。

Terminate：请求结束 DDE 会话通道。

Request：客户请求从服务器上获取数据。

Advise：客户请求从服务器上获得更新的数据。

Poke：客户请求服务器改变数据的值。

Execute：客户请求服务器执行命令。

3 监控程序的实现

Windows DDE 功能应用的核心是 DDEML（Dynamic Data Exchange Management Library 动态数据交换管理库），负责管理 Windows 操作系统中应用程序间的会话和通信。DDEML 管理 DDE 通信的基础是 Windows 消息机制，DDE 通信的实质是在客户端和服务器之间传送 DDE 消息，直接使用 DDE 消息机制设计开发具备 DDE 功能的客户/服务器应用程序，虽然工作原理非常简单，但很难保证 DDE 功能的完整性及整个程序的健壮性，如 DDE 消息响应函数中对参数类型数据指针的处理非常敏感，如处理不当会引发程序错误。

DDEML 可以克服直接采用 DDE 消息机制进行程序设计的缺陷，DDEML 将数量多，难以记忆和理解的 DDE 消息及参数加以封装，以 API 函数的形式提供给用户使用，降低了设计难度，保证了 DDE 功能的完整性和健壮性，是开发 DDE 应用程序的首选方案。使用 DDEML 后，实际上客户和服务器之间的多数会话并不是直达对方的，而是经由 DDEML 中转，即用 Callback 函数处理，而早期的消息通信是直接的。

客户端监控程序的编程平台选用微软公司的 Visual C++ 6.0，其具体的实现步骤如下：

（1）首先要包含头文件 #include “ddeml.h”，以正确使用 DDEML 的 API 函数。

（2）全局变量的定义。主要包括会话句柄、LNS DDE Server 的程序名、主题名和项目名字符串句柄。定义如下：

```
DWORD   idInst=0;  //DDEML 实例 ID
HSZ     hszApp=0;  //Server 程序名字符串句柄
HSZ     hszTopic=0;//Server 主题名字符串句柄
HSZ     hszItem[DDE_ITEM_NUMS];//Server 项目名字符
//串句柄
```

（3）打开 LNS DDE Server。只有在 LNS DDE Server 完全打开并初始化成功后，才能保证客户端的监控程序的 DDE 初始化成功，需要利用函数 Sleep () 惊醒延时后，进入下一步。

（4）程序初始化，包括 DDE 初始化、创建程序名（Application）、主题名（Topic）和项目名（Item）；与 LNS DDE Server 建立连接，并设定为热连接型的会话模式，保证设备上的数据发生更新及时上报给客户端监控程序；注册服务。具体实现封装成函数如下：

```
BOOL CMonitorView::InitDdeClient()
{
 //1)初始化
 UINT iReturn;
   iReturn = DdeInitialize( &idInst,  //实例的 ID 指针
      (PFNCALLBACK)DdeCallback,  //回调函数指针
                                  APPCLASS_STANDARD
| APPCMD_CLIENTONLY,  //初始化参数
      0 );                        //保留参数(置 0)
   if (iReturn! =DMLERR_NO_ERROR)
   {
      AfxMessageBox("DDE 初始化失败");
      return FALSE;
   }
  //2)创建 DDE string
hszApp=DdeCreateStringHandle(idInst,DDE_SERVER_APP,0);
                  hszTopic=DdeCreateStringHandle(idInst,
DDE_SERVER_TOPIC,0);
  for(int i=0;i<DDE_ITEM_NUMS;i++)
 hszItem[i]=DdeCreateStringHandle(idInst,pszItem[i],0);
  //3)注册服务
  DdeNameService(idInst,hszApp,0,DNS_REGISTER);
  //建立连接
  Sleep(2000);//不延时会报"DDE 连接失败！"
   hConv=DdeConnect(idInst,hszApp,hszTopic,NULL);//连
//接服务端
  if(hConv)//如果建立成功
  {
      DWORD dwResult;
  for(int i=0;i<DDE_ITEM_NUMS;i++)//生成热连接型会话
  DdeClientTransaction    (NULL,0,hConv,hszItem    [i],
CF_TEXT,XTYP_ADVSTART,TIMEOUT_ASYNC,&dwResult);
  }
  else
      AfxMessageBox("DDE 连接失败！");
  return TRUE;
}
```

（5）DDE 回调函数的处理。根据服务器的响应，按照项

目名的不同进行不同的处理。是整个监控程序的实现核心。实现的函数如下：

```
HDDEDATA CALLBACK DdeCallback(//回调函数
                    UINT uType,    // 传输类型
                UINT uFmt,     // 剪切板数据格式
                    HCONV hConv,   // 会话句柄
                HSZ hTopic,    // 主题字符串句柄
              HSZ hItem,     // 项目名字符串句柄
              HDDEDATA hData, // 全局内存对象句柄
            DWORD dwData1,  // 指定的传输数据
              DWORD dwData2)  //指定的传输数据
{
 int i;
 char tmp[255];
 CString str;
 switch(uType)
 {
 case XTYP_ADVSTART:
 case XTYP_CONNECT:   //连接请求
      return (HDDEDATA)TRUE;
 case XTYP_ADVDATA:   //有数据到来
      for(i=0;i<DDE_ITEM_NUMS;i++)
      {
           if(hItem==hszItem[i])//确定数据项
DdeGetData(hData,(PBYTE)tmp,255,0);//取得数据
           switch(i)
           {
           case 0: case 1: case 2: case 3:
                fTurnout[0][i]=atof(tmp);
                break;
           case 4: case 5: case 6: case 7:
                fTurnout[1][i-4]=atof(tmp);
                break;
           }
      }
      return (HDDEDATA)DDE_FACK;//回执
 }
 return 0;
}
```

（6）DDE 退出。包括注销服务，注销初始化和释放程序名（Application）、主题名（Topic）和项目名（Item）。

6 结语

采用 Windows DDEML 实现 LonWorks 网络监控程序，原理清晰简单，开发周期短，上手容易，效率高，能满足监控程序的通信要求。

参考文献

[1] 高安邦，等. LonWorks 技术开发和应用. 北京：机械工业出版社，2009.

[2] 高安邦，杨帅，陈俊生. LonWorks 技术原理与应用. 北京：机械工业出版社，2009.

[3] 周江. 利用 VC 实现 DDE 服务客户应用. 电脑编程技巧与维护，2003，2.

（收稿日期：2012-10-24）

（上接第 70 页）

首先在地图内画一个多边形区域，如图 12 所示。

图 12

然后分别点击“show data”、“show color”按钮，效果如图 13 所示。数据和颜色分别都显示出来了。

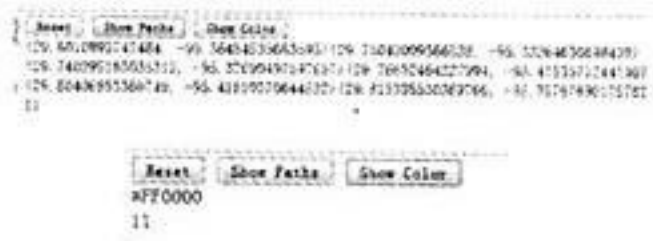


图 13

5 结语

Google MAP API 所提供的接口，丰富了普通网站页面的内容，为网站提供了免费的地理图形显示功能，在此基础上又提供了多种地图应用，方便了用户和网站的交互。

通过 GoogleMaps 为开发者提供的地图 API，可以开发出各种各样有趣的地图 Mash-up 应用，还可以将不同地图图层加载到应用中，如卫星影像、根据海拔高度绘制的高山和植被地形图、街道视图等，从而帮助开发者打造个性化的地图应用站点。

地图 API 是一项免费的服务，任何非盈利性网站均可使用。

参考文献

[1] 《Google map api v3》. Http://developer.google.com/.

[2] 《Google map api》. Http://www.google.com/apis/maps/.

（收稿日期：2012-09-05）

TROUBLESHOOTING OF PROGRAM

Q 怎么把 Word 文档转成 PPT 幻灯片

A 首先生成一个对话框应用程序，编译后没错。在 Win7 下开发，建议在头文件 stdafx.h 的最顶端加入：

```
#define WINVER 0x0700
#define _WIN32_WINNT 0x0700
```

然后点击“类视图”标签，添加操作 PPT 和 Word 文档所需要的类。其中 PPT 的类有 13 个，Word 类 7 个。以添加 PPT 类为例，具体方法如下：右单击“类视图”中的工程名，选择“添加”→“类”，在弹出的对话框中选择“TypeLib 中的 MFC 类”，按“添加”按钮，在弹出的对话框中选择“Microsoft PowerPoint 14.0 Object Library<2.0>”，选择左侧列表框中的类，它们是：_Application，_Presentation，Presentations，_Slide，Slides，TextFrame，TextRange，Shapes，Shape，ShadowFormat，ColorFormat，Font，SlideShowTransition。添加 Word 类的方法相似，选择“Microsoft Word 14.0 Object Library<8.5>”，它们是：_Application，_Font，_Document，Documents，Bookmarks，Bookmark，Selection，Range，Bookmarks，Bookmark。可以看到这两套类中有几个类的名字重合了，为了方便后面引用，可以在相应的类集合中添加名字空间，比如对于 PPT 类，使用：namespace PPTSPACE，对于 Word 类使用：namespace WordSpace。还有一点需要注意，就是必须把这 20 个类的第一行注释掉，否则会出现大量编译错误，多至上百个。

这一行在 PPT 中是：

```
#import"E:\\Microsoft Office\\Office14\\MSPPT.OLB"no_namespace
```

这一行在 Word 中是：

```
#import"E:\\Microsoft Office\\Office14\\MSWord.OLB"no_namespace
```

做完以上辅助工作，下面可以使用这些接口操作 PPT 与 Word 文档的读写了。第一步先生成操作 PPT 和 Word 文档的实例，分别使用这两个 Application 对象的 CreateDispatch 方法来创建，代码如下：

```
PPTSPACE::CApplicationm_PPT;
WordSPACE::CApplicationm_WRD;
m_PPT.CreateDispatch("Powerpoint.Application");
m_WRD.CreateDispatch("Word.Application");
```

然后连接 PPT 模板实例，在 Open 方法中的第二个参数值为 0，表示拥有读写操作权限，剩下参数的值都为-1，表示忽略其行为，代码如下所示：

```
PPTSPACE::CPresentationsm_PresSet;
PPTSPACE::CPresentationm_PresItem;
m_PresSet.AttachDispatch(m_PPT.get_Presentations());
m_PresItem=m_PresSet.Open("tmp\\模板.ppt",0,-1,-1);
```

接下来连接 Word 文档实例，它的 Open 方法参数很多，但大多数都是可选的，具体可查看 MSDN 文档，代码如下：

```
COleVariantCovOpt ((long)DISP_E_PARAMNOTFOUND,
VT_ERROR);
COleVariantCovTrue((long)1),CovFalse((long)0);
PPTSPACE::CSlidesm_SldSet;
WordSPACE::CDocumentsm_DocSet;
WordSPACE::CDocumentm_DocItem;
m_SldSet=m_PresItem.get_Slides();
m_DocSet=m_WRD.get_Documents();
m_DocItem.AttachDispatch (m_DocSet.Open (COleVariant (L"
演 示 .doc",VT_BSTR),CovFalse,CovFalse,CovFalse,CovOpt,
CovOpt,CovFalse,CovOpt,CovOpt,CovOpt,CovOpt,CovTrue,
CovOpt,CovOpt,CovFalse,CovOpt));
```

下面可以把 Word 文档按行读出来。一般 Word 文档中的内容主要是表格、图片、文本，所以这里也只处理这 3 种对象。先罗列整体框架，后面再分别处理，代码如下：

```
WordSPACE::CSelectionm_Sel;
m_TmplNum=m_SldSet.get_Count();
m_Sel=m_WRD.get_Selection();
int line(m_DocItem.ComputeStatistics(1,CovFalse));
CStringWtxt(""),prev(""),para("");
int tag(-1),page(1),word(0);
CStringtip("");
for(int i=1;i<=line;++i)
{
    m_Sel.GoTo (COleVariant (3L),COleVariant (1L),
COleVariant((long)i),
    COleVariant("",VT_BSTR));
    switch(tag)
    {
            case 1: //表格
                i=CopeTable(i,tag,page);
                break;
            case 2: //图片
                i=CopePictrue(i,tag,page);
                break;
            default: //文字
        i=CopeText(i,tag,page,word,txt,prev,para);
                break;
    }
}
```

这段代码的用意是先用 Application 对象中的 get_Selection() 方法获取 Selection 对象，用于选择文档中的内容。再使用 Document 对象中的 ComputeStatistics 方法得到文档总行数，参数值 1 表示 wdStatisticLines，然后根据不同情况处理。为了方便确定对象类型，预先在 Word 文档中，每个表格和图片前加上提示标识。规定“表”和“图”两个汉字后面，紧跟一个任意的阿拉伯数字即可，这样可以用正则表达式进行匹配搜索。先来看一下读取表格的处理，代码如下：

```
intCopeTable(inti,int&tag,int& page)
{
    COleVariantrow((long)0);
    m_BkmkSet=m_DocItem.get_Bookmarks();
m_BkmkItem =m_BkmkSet.Item (COleVariant ("\\Table",
VT_BSTR));
    m_Rng=m_BkmkItem.get_Range();
    m_Rng.Copy();
    row=m_Rng.get_Information(15);
    i+=row.lVal-1;
    (m_TmplNum>1)?InsertObject2 (page++):InsertObject1
(page++);
    tag=-1;
    return(i);
}
```

为了定位表格，先用 Document 对象的 get_Bookmarks 方法得到标签集合 Bookmarks，再通过预定义类型“\Table”得到当前标签下的表格对象，最后用 Range 对象的 Copy 方法把整个表格复制到剪贴板，以便后面插入 PPT 文档中。返回前用 Range 对象的 get_Information 方法得到表格最大行数，参数值 15 表示 wdMaximumNumberOfRows，以便继续读取后面的内容。对图片处理类似，只是简单一些，代码如下：

```
intCopePictrue(inti,int&tag,int& page)
{
    m_Sel.HomeKey(COleVariant(5L),COleVariant(1L));
    m_Sel.EndKey(COleVariant(5L),COleVariant(1L));
    m_Rng=m_Sel.get_Range();
    m_Rng.Copy();
    (m_TmplNum>1)?InsertObject2 (page++):InsertObject1
(page++);
    tag=-1;
    return(i);
}
```

其中 Selection 对象的 HomeKey 方法和 EndKey 方法起到选择本行的作用，第一个参数值 5 表示 wdLine，第二个参数值 1 表示 wdExtend。接下去的操作都是相同的，不再赘述了。对文本的处理稍微复杂一些，由于 PPT 文档的演示面积有限，超出其范围的文字就看不见了，又不能出现滚动条，所以需要自己切分 Word 文档中大段的文本，代码如下：

```
intCopeText(inti,int&tag,int&page,int& word,
    CStringW&txt,CStringW&prev,CStringW&para)
{
    m_Sel.HomeKey(COleVariant(5L),COleVariant(1L));
    m_Sel.EndKey(COleVariant(5L),COleVariant(1L));
    prev=txt;
    m_Rng=m_Sel.get_Range();
    txt=m_Rng.get_Text();
    if(prev! =L'\r'&&prev==txt) i=int(9E8);
    else
    {
        intlen(txt.GetLength());
        if(len+word>m_MaxWord)
        {
            if(word>0)
            {
(m_TmplNum>1)?InsertText2(para,page++)
:InsertText1(para,page++);
                word=len;
            }
            para=txt,word=len;
            if(word>m_MaxWord)
            {
                CStringWpart("");
                int j(0);

                for(;j<word/m_MaxWord;++j)
                {
    part=para.Mid(i*m_MaxWord,m_MaxWord);
    (m_TmplNum>1)?InsertText2(para,page++)
:InsertText1(para,page++);
                    word-=m_MaxWord;
                }
                para=para.Mid(j*m_MaxWord,word);
            }
        }
        else
        {
            para+=txt;
            word+=len;
        }
        tag=AnalyzeString(txt,i);
        if(tag>0)
        {
(m_TmplNum>1)?InsertText2(para,page++)
:InsertText1(para,page++);
para.Empty();
            word=0;
        }
    }
    return(i);
}
```

其中的 m_MaxWord 可根据 PPT 文档模板尺寸自行设定，这里为 100 字。注意到辅助函数 AnalyzeString，它就是用来分析本行内容是提示标识，或是其他文字。主要是两个正则表达式的搜索模板，对于表格，其模板为：\\w* (表 [0-9] +) \\w*，而对于图片，其模板为：\\w* (图 [0-9] +) \\w*，具体内容可参考所附例程的代码。

下面看一下如何把得到的内容写入 PPT 文档中，即上面代码里的 InsertXXX 系列辅助函数。这里根据所选择的 PPT

模板和不同的对象，做相应的处理。第一种情况，对于表格和图片，如果模板只有一页，只需简单添加即可，代码如下所示：

```
void InsertObject1(int page)
{
    m_SldItem=m_SldSet.Add(page,12);
    m_ShpSet=m_SldItem.get_Shapes();
    m_ShpSet.PasteSpecial(10,0,0,0,0,0);
}
```

其中 Shapes 对象的 Add 方法中，第一个参数表示页码，第二个参数值为 12 表示 ppLayoutBlank，方法 PasteSpecial 则黏贴具体对象，第一个参数值为 10 表示 ppPasteOLEObject。如果模板有多页，那么先重新增加一组模板，然后定位到当前位置，再写入具体内容：

```
void InsertObject2(int page)
{
    m_TmplNum=m_SldSet.get_Count();
    if(0==page%m_TmplNum)
    {
m_SldSet.InsertFromFile ("tmpl\\ 模 板 .ppt",page,1,
m_TmplNum);
    }
    m_SldItem=m_SldSet.Item(COleVariant((long)page));
    m_ShpSet=m_SldItem.get_Shapes();
    m_ShpSet.PasteSpecial(10,0,0,0,0,0);
}
```

其中 Shapes 对象的 get_Count () 方法得到文件中的模板页数，InsertFromFile 方法从文件中添加一组模板，注意索引是从 1 开始。再用 Item 定位到当前位置，后面的代码都相同。第二种情况，对于纯文本内容，若模板只有一页，只需将 Shapes 对象的 Add 方法中，第二个参数的值改为 2 即可，表示 ppLayoutText。若模板有多页，定位方式相同，只是在写入 PPT 文档时有所区别，代码如下：

```
voidDisplayText(CStringW& txt)
{
    USES_CONVERSION;
m_ShpSet=m_SldItem.get_Shapes();
m_ShpItem=m_ShpSet.AddTextbox(1,m_BoxLeft,m_BoxTop,
    m_BoxWidth,m_BoxHeight);
    m_TxtFrm=m_ShpItem.get_TextFrame();
    m_TxtRng=m_TxtFrm.get_TextRange();
    m_TxtRng.put_Text(W2A(txt));
    m_Font=m_TxtRng.get_Font();
    m_Font.put_Size(1.0f*m_FontSize);
    m_Font.put_Name(m_DocFont.lfFaceName);
    m_Font.put_Bold((m_DocFont.lfWeight>400)?1:0);
    m_ClrFmt=m_Font.get_Color();
    m_ClrFmt.put_RGB(RGB(m_Rval,m_Gval,m_Bval));
    m_ShdFmt=m_ShpItem.get_Shadow();
    m_ClrFmt=m_ShdFmt.get_ForeColor();
    m_ClrFmt.put_SchemeColor(2);
    m_ShdFmt.IncrementOffsetX(4);
    m_ShdFmt.IncrementOffsetY(4);
    m_ClrFmt.put_SchemeColor(3);
    m_ClrFmt.put_RGB(RGB(250,30,50));
    m_ShdFmt.put_ForeColor(m_ClrFmt);
    m_ShdFmt.put_Visible(-1);
}
```

主要操作是前面几行，先用 Shapes 对象的 AddTextbox 方法添加文本框，第一个参数的值为 1 表示 msoTextOrientationHorizontal，然后用 TextRange 对象的 put_Text 方法写入内容。剩下的代码是使用 Font 对象设置字体的属性，使用 ColorFormat 对象设置字体颜色，使用 ShadowFormat 对象设置字体阴影，可根据需要做相应调整。唯一值得注意的是，在设置阴影颜色时，几条语句的顺序不能改变，否则看不到效果。

在 PPT 幻灯片生成以后，还可以在幻灯片之间加入不同的转换特效。只要使用 SlideShowTransition 对象的 put_EntryEffect 方法即可实现，然后使用 Presentation 对象的 SaveAs 方法保存整个文档内容。在退出应用前，使用每个对象的 ReleaseDispatch 方法释放内存，具体可参考所附例程。

由于 Word 文档千差万别，本文只实现了对 3 种基本对象的处理，而且对表格和图片的处理还需要预先加上提示标识，并不完美，转换后的 PPT 文档在视觉效果上与专业制作也有差距，还需要进一步完善，希望和大家共同探讨。所附例程在 VS2010+Win7 64 位下调试通过，至少需要安装 Office 2010 中的 Word 和 PowerPoint 组件。后面是 MSDN 里的一些资源，罗列在此，方便大家查阅。

（作者：申晓）

（上接第 87 页）

3 结语

通过测试，软件能很好地执行关机提示和自动关机功能，软件的安装应用能强制电脑定时自动关机，对节约电能资源、减少计算机无谓的损耗都有一定的实际应用价值。

参考文献

[1] （美）沃森（Watson,K.），（美）内格尔（Nagel,C.），等. C# 入门经典 4 版. 齐立波，译. 清华大学出版社，2008，12.

（收稿日期：2012-08-06）

电脑系统维护经验与技巧

怎样打造硬盘维护管理工具箱

硬盘的擦除、分区、备份、检测、修复等操作在日常电脑使用中经常遇到，都需要运用专门的工具，现在打造一个集众多硬盘工具于一身的闪存，这样管理硬盘将会非常轻松。

（1）打造硬盘管理闪存

首先将一个 1GB 以上容量的闪存插入电脑，下载 Parted Magic 和 Unetbootin 两款工具。运行 Unetbootin-w in-549.exe，在其主界面勾选“Diskimage”，浏览选择 pmagic-6.3iso 文件，再选择闪存的启动类型和盘符，然后点击“OK”。

这样闪存将被装入一个类似 WinPE 的可启动微系统，但它基于 Linux，需要将电脑的 BIOS 设为闪存启动，在启动后会出现一个选单，选择选项 1：“Default Default settings (Runs from RAM) ”就进入了图形界面的 Parted Magic 平台。

在 Parted Magic 中包含了非常多的硬盘管理以及其他软件，由于是在内存中运行（内存需要 256MB 以上），不会调用电脑硬盘，因此它们可以直接对硬盘进行更全面的操作，当然也可以一边格式化硬盘一边上网听歌。值得一提的是，它的硬件要求非常低，适合在处理器为奔腾以上级别的电脑上使用。

（2）管理硬盘多面手

Parted Magic 内部集成的工具非常多，下面来试试几个比较实用的。

1）轻松擦除 SSD

SSD 和传统硬盘不同，随着使用时间一长，性能会下降，这时需要擦除一下恢复到出厂状态。通常会使用 HDDErase，但它需要在 DOS 模式下使用，而使用 Parted Magic 中的 Secure Erase 就要简单一些。点击“System Tools”→“Erase Disk”，在弹出界面中勾选“Intem al：Secure Erase command writes zeroes to entire data area”项目，点击下一步，选择当前 SSD，点击“OK”。

如果出现“frozen”（冻结）的的信息，点击“Sleep”按钮将电脑进入睡眠状态，然后唤醒系统，再重复前面的操作。

接着将密码设为“NULL”点击“OK”，这时会出现同上警告信息，点击“Yes”即可进行擦除工作。需要注意的是在进行操作的时候，不要将装有 Parted Magic 的闪存拔离电脑。

2）修改硬盘系统密码

忘记密码是用户普遍的问题，如果忘记了硬盘中 Windows XP/Vista/7 的登录密码，那么用户可以通过 chntpw 来进行清除。

3）全面的克隆工具

Clonezilla 是一个很好的系统克隆工具，它不仅支持对整个系统进行克隆，而且也可以克隆单个硬盘分区，这种灵活性更能适应用户的备份需要。Clonezilla 的最大特点是支持的格式非常丰富，既可以克隆 Linux 系统，也能够对 Windows 系统进行克隆。Clonezilla 还支持使用 PXEBoot 来进行 Multicast 克隆，这对于需要克隆大量系统的用户来说极为重要。

其实 Parted Magic 的用途还有很多，大家挨个去试试就知道它们的作用了。俗话说有备无患，制作这样一个闪存在手上，无论何时何地都可以对电脑的硬盘进行全面的管理。

硬盘在操作系统中的实际容量是多少

关于硬盘标称容量和系统显示不符已经是老生常谈的问题了，其实问题关键就在于操作系统厂商和硬盘厂商在换算标准上不统一，操作系统为 1024 字节=1KB，1024KB=1MB，1024MB=1GB，而硬盘厂商则是整数换算标准，即 1000 字节=1KB，1000KB=1MB，1000MB=1GB，照此换算下来，硬盘显示容量总和是正常的。关于测试硬盘的软件，推荐 HD Tune。HD Tune 不仅能检测磁盘信息，还能够测试性能，检测磁盘健康状况。

硬盘检测软件 HD Tune 值得关注的功能是什么

最常用的硬盘检测软件莫过于 HD Tune 了，这款小巧的软件不仅能测试硬盘的读写性能，还能让用户直观地了解到硬盘的健康状况并扫描错误，全方面掌握硬盘信息。其常用的几项功能如下：

（1）检测信息

这一栏主要是检测选定硬盘的基本信息，指明了序列号、固件版本、缓存、分区情况、支持特性等，如同 CPU-Z 一样，有助于用户对硬盘有概括性认识。

（2）健康和错误扫描

硬盘的“健康”直接关系到数据的安全和用户使用中体验，也是大家非常关心的一项。在健康这一栏中，首先可以了解到硬盘的通电时间，这对新硬盘用户比较重要。刚上机的硬盘用电时间应该为 0（以小时计算），以此可以直接判断硬盘是否为翻新品或维修过的。

在界面里用户能看到很多与硬盘健康相关的具体参数，其中“阈值”代表临界值，“数据”代表当前硬盘状况，出现问

题或警告的以黄色或红色横条标出。不少用户在论坛里发帖担心硬盘出现“三黄”就是指 HD Tune 健康检测中有 3 项出现警告。要注意的是软件的颜色指示并不完全准确，很多新硬盘在 HD Tune 检测下都会出现“两黄”现象，而故障的数据显示都是 0。因此不可迷信软件的颜色指示，而要看关键健康参数情况。

最重要的便是重映射扇区计数，重映射扇区是硬盘厂商提供的备用扇区，如果用户在使用过程中出现了损坏的区域，那么备用的扇区就会将其替换。不过这部分扇区毕竟是有容量限制的，如果达到了阈值，硬盘坏道便随之产生了。因此在 HD Tune 健康检测中，“重映射扇区计数”这一项为黄，并且“数据”一项不断增长，就要引起足够重视了。硬盘已处在亚健康状态，最好将重要数据加以备份以避免损失。

其余的健康参数不需过多留意，一般只要重映射扇区计数为零，且健康状态为良好或好，硬盘就比较安全的。对于那些重映射扇区计数较高，或者出现红色警告，健康状态为衰退的硬盘，有必要在“错误扫描”一栏进行坏道检测。一旦检测情况不理想，就要迅速备份并修复了。

(3) 基准测试

磁盘性能是用户比较关心的一项，HD Tune 里主要是检测硬盘的读写性能（硬盘中没有数据才能进行写入测试），测试前需要关闭其他软件以免造成影响。

点击“开始”后便开始检测，数据会以折线表示出来，方便观察整体情况。这里需要注意的是“平均传输速率”和“突发传输速率”，直接反映了硬盘性能情况。此外，“最低传输速率”也是很重要的参数，如果数值过低表明传输不稳定，实际应用中会出现“突然一卡”的现象，也是硬盘衰退的表象之一。此外“文件基准”和“随机存取”测试更接近于实际应用，也有一定参考价值。常用的速率检测软件还有 ATTO Disk Benchmark、HD Tach 等，测试的侧重点不同，对用户全面了解磁盘性能也有帮助。

? 使用磁盘工具软件要注意什么

! 俗话说软硬不分家，稳定并有效地使用硬件，离不开软件的支持和维护。通过磁盘软件，一方面有助于用户对硬盘的安全和性能情况有详细的了解，另一方面也有助于加快磁盘传输、提升工作效率。同时在日常应用中，还要注意以下几点以提升硬盘的稳定性，并优化性能。

(1) 使用较高质量的传输线材，并保证供电稳定（多硬盘用户更需留意）。

(2) 对于“高温”硬盘，有必要进行辅助降温，以延长寿命（可采用加装风扇和散热片等方式）。

(3) 避免频繁开关机，硬盘工作时不要轻易移动，定期清理机箱，避免灰尘堆积。

(4) 使用 P2P 下载软件（如 BT、迅雷、电驴等），将下载缓存调至较高水平，减少对磁盘损害。有条件的用户还可选定专门的硬盘做下载之用（如西部数据的绿盘）。

? 如何通过右键快速整理磁盘

! Windows 系统磁盘整理功能总是需要右键选中硬盘分区，在弹出的菜单中点击“属性”→“工具”，然后才能调用磁盘整理工具，操作过程显得有点过于麻烦了。其实通过一个小技巧就能将磁盘整理工具放入右键菜单，右击任意硬盘分区就可以马上进行磁盘整理了。

首先，按下 Win+R 键，调出独立运行窗口，键入“regedit”打开注册表编辑器，接下来点击“HKEY_CLASSES_ROOT\Drive\shell”选项，然后在左侧面板中新建项目，命名为“run as”。将右侧面板中的 Default 值修改为“Defragment”，随后再在右侧面板中新建名为“Extended”的键值即可。回到磁盘管理窗口中，右键点击任意磁盘分区，在弹出的菜单中就可以看到“Defragment”（磁盘整理）。

当然还要注意用户权限的问题，需要使用管理账号。如果觉得修改注册表不方便，可以直接点击提供的注册表文件。

? 如何打开磁盘清理程序的一些选项

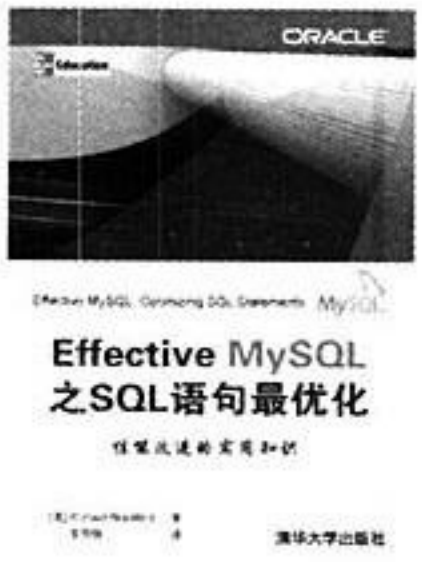
! Windows 自速写带的“磁盘整理”功能可以让用户清理一些不必要的垃圾文件，让系统能够腾出更多空间，不过这个功能的清理选项比较少，无法更多地清理一些不必要的文件。用户可以在原来的磁盘清理界面中看到，它只能清理基本的缓存区以及回收站，其实还有很多隐藏的垃圾文件没被清理掉。

因此用户可以打开磁盘清理程序的一些隐藏选项，将系统清理得更干净彻底。按下 Win+R 组合键调出“运行”对话框，输入“cleanmgr/sageset.99”,并点击“确定”按钮，接着就会出现磁盘清理的选项设置界面，它比原来的磁盘清理多了很多项目。

勾选想要清理的选项，然后点击“确定”按钮。再次调出“运行”对话框输入“cleanmgr/sageset.99”,按下“确定”，就可以开始清理了。当清理完成后，重新启动电脑，就会发现硬盘分区的剩余空间变得更大一些了。

? 笔记本固态硬盘如何在台式电脑上使用

! 只要是采用了标准的 SATA 接口，笔记本固态电脑也可以很容易地使用在台式电脑上。只需要正确连接，并把固态硬盘设置为系统盘，将机械硬盘设置为数据盘，系统的性能就会获得明显的提升。不过需要注意的是，很多固态硬盘需要在 AHCI 模式下才能发挥全部性能，所以别忘了在 BIOS 中进行相应的设置。此外，台式电脑的机箱通常没有 2.5 英寸或 1.8 英寸的驱动器安装位，所以还需要根据固态硬盘的尺寸购买一个转接架，才能稳妥地使用笔记本的固态硬盘。



书名：Effective MySQL 之 SQL 语句最优化
ISBN：978-7-302-30429-6
定价：29 元
作者：(美) 布拉德福 (Bradford, R.) 著

《Effective MySQL 之 SQL 语句最优化》是由 MySQL 专家 Ronald Bradford 撰著，书中提供了很多可以用于改进数据库和应用程序性能的极佳实践技巧，并对这些技巧做了详细的解释。本书通过一步步详细介绍 SQL 优化的方法，帮助读者分析和调优有问题的 SQL 语句。

书中讲述了找出收集和诊断问题必备的分析命令；创建 MySQL 索引来改进查询性能；掌握 MySQL 的查询执行计划；找出影响查询执行和性能的关键配置变量；用 SQL 语句优化的生命周期来识别、确认、分析然后优化 SQL 语句，并检查优化的结果；学习使用不为常人所知的一些性能技巧，来改进索引效率并简化 SQL 语句。

书名：Red Hat Enterprise Linux 系统管理
ISBN：978-7-302-30449-4
定价：48.00 元
作者：朱居正 编著

本书以 Red Hat Enterprise Linux Server release 6 (Santiago) 为蓝本，全面介绍了 Linux 的基本概念、特点、重要安装步骤、GNOME 桌面管理、系统各项基本配置、文件系统、用户管理、系统管理、网络管理、磁盘管理、Linux 常用命令、Shell 编程、DNS、DHCP、FTP、NFS、Samba、Web 等各种服务器架设、NAT、VPN、LDAP、VNC 和 Openssh 应用、Linux 系统的安全设置等内容。

书中结合实际与实践应用，讲解具体，操作性强。无论是简单的 Linux 命令、复杂的服务器配置与管理，还是语言编程和安全管理，都采用通俗易懂的语言并配以简单明了的图片进行介绍，力求把复杂的问题具体化、简单化、明了化。同时，本书还穿插了笔者多年来在实际应用 Linux 过程中积累的大量实例。

书名：Oracle Fusion Middleware 11g 架构与管理
ISBN：978-7-302-30425-8
定价：50.00 元
作者：(美) 夏菲 (Shafii, R.)，(美) 李 (Lee, S.)，(美) 肯杜瑞 (Konduri, G.) 著

通过学习书中提供的丰富信息，掌握敏捷、智能业务应用程序的统一平台。本书完整讲解了 oracle fusion middleware 组件，并展示了核心用例、实践和逐步分析的管理指南。您将学到如何提供服务器和群集、配置 web 服务、管理门户和优化 oracle fusion middleware 组件的性能。另外，监视、诊断和安全也将在这本权威的图书中进行讨论。

本书特色理解 oracle fusion middleware 11g 的关键架构概念；创建和部署 oracle weblogic server 的域和群集；设置和管理使用 oracle application development framework 构建的应用程序；充分发挥 oracle soa suite 环境的价值；管理 oracle webcenter 中的门户和 enterprise 2.0 服务。

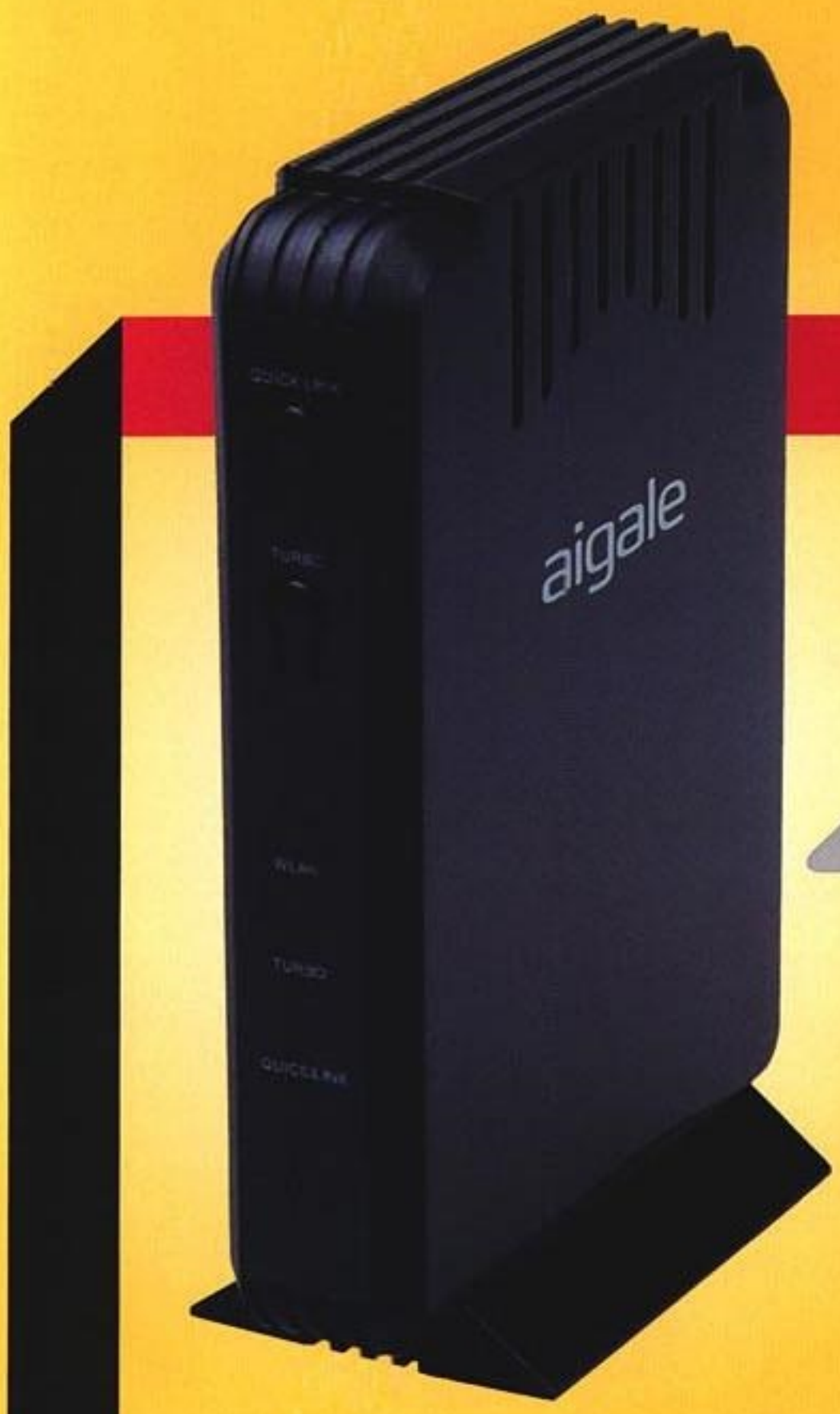
书名：移动金融：创建移动互联网时代新金融模式
ISBN：978-7-302-30561-3
定价：45.00 元
作者：李麟 钱峰著

本书立足于我国商业银行发展移动金融的现实背景，立足于浦发银行与中国移动的战略合作，将研究重点放在移动金融发展的商业模式、客户定位、产品创新和渠道经营等四大方面，提出了我国商业银行移动金融发展的关键在于全面创新。本书旨在通过论述这四个主要领域的不断创新，推动我国商业银行移动金融的快速发展。

本书是国内移动金融领域前沿巨作，作者李麟为浦发银行战略发展部总经理，撰写移动金融专著，具有很高的权威性。本书由浦发银行董事长吉晓辉作序推荐，书中既有理论深度，又结合了国内商业银行的实际，系统化地提出了现阶段我国商业银行移动金融的发展策略。立足移动金融的现实，阐述移动新金融模式的发展。

抢先Hold住PCWorld

即可精巧“联”通科技未来！



现在邮购2013年
《微电脑世界》**全年杂志**

即得一个价值149元

海联达Ai-R100 极风
无线路由器

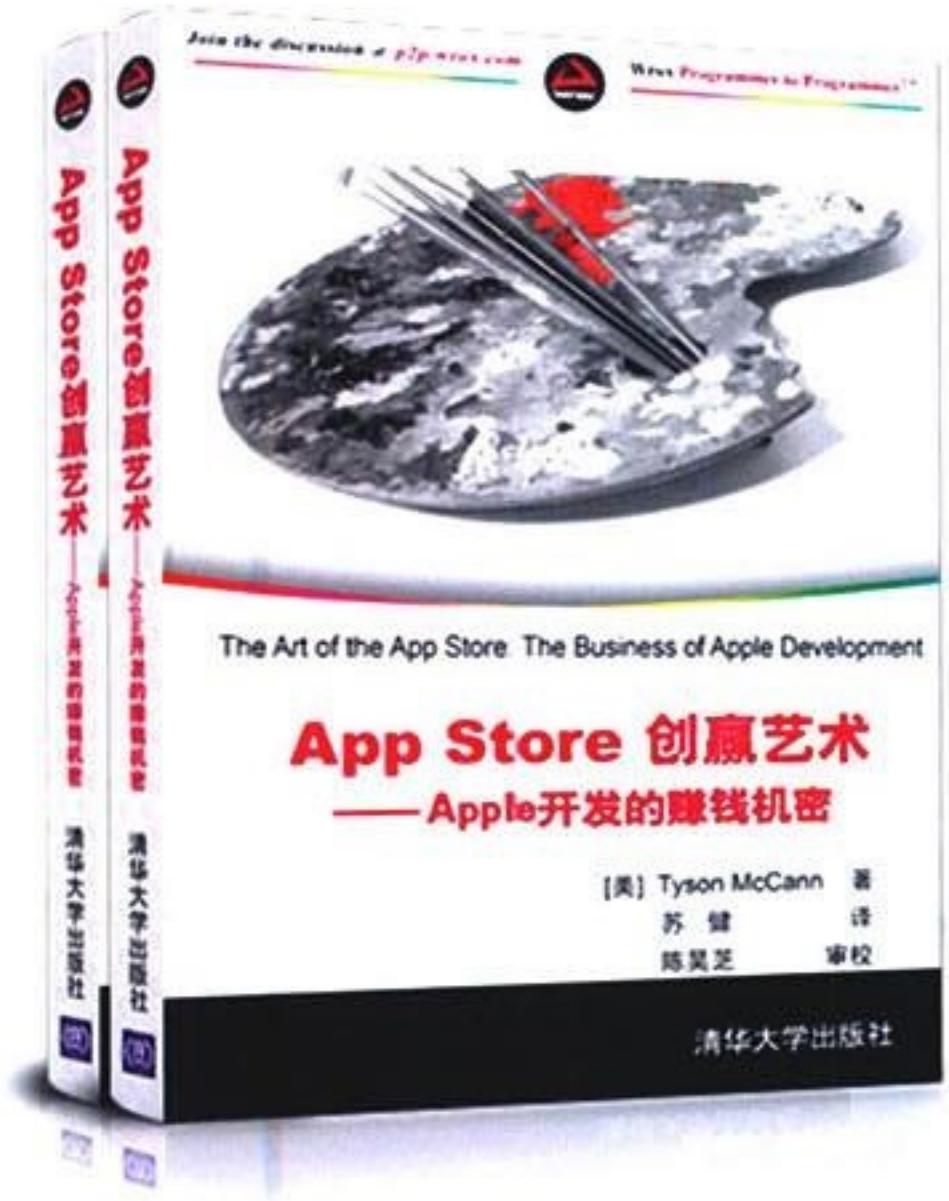
轻松联通您的智能终端，
让您尊享全球IT资深顾问
随时随地的贴身资讯服务。

汇款地址：北京市123信箱，收款人：微电脑世界杂志，邮编：100036
杂志定价：144元/年（12元/月）
活动咨询：周一到周五，9：00~11：30，13：00~17：30
电话：010-63130909-1829
杂志社现场订阅地址：北京市海淀区万寿路翠微中里14号楼
在线订阅：http://www.pcworld.com.cn/about/ebuy/pay.html

活动说明：

活动时间：2012年8月10日~2013年3月31日（邮局汇款以邮戳为准）
在汇款单附言栏注明“2013年微电脑世界”，同时留下联系电话。
如需发票，请在汇款单附言注明“发票”以及发票抬头，过后将不能补开。
本活动仅限于在杂志社订阅的读者，邮局订阅等其他渠道不参加此活动。
由于本次活动涉及奖品发放，参与活动的读者将不能中途退订。
邮费：平寄邮费由杂志社承担，如需挂号，每本另加3元挂号费，汇款时一并汇上，并注明挂号字样。

在“狂野西部”中淘金 App Store创赢艺术



App Store创赢艺术——Apple开发的赚钱机密

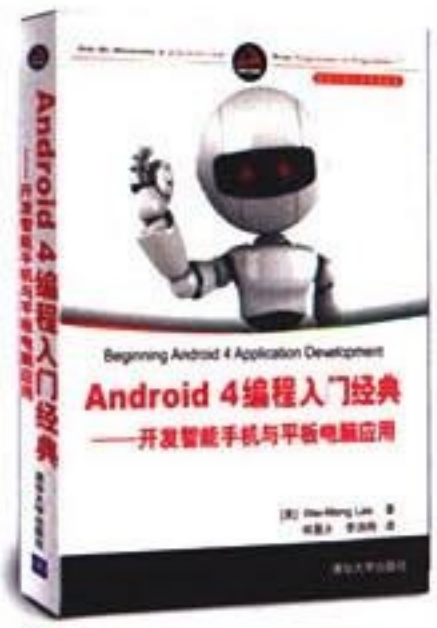
作 者：［美］Tyson McCann

ISBN：978-7-302-30501-9

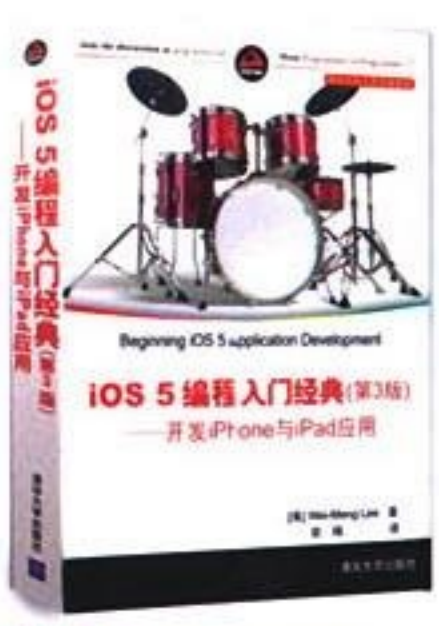
定 价：59.00元

- 保证自己的应用能在竞争中脱颖而出，大赚一笔！
- 一本商业指南
- 亚马逊五星图书
- App Store目前和未来的趋势
- 一站式学习规划、创建、营销和维护应用程序!!!

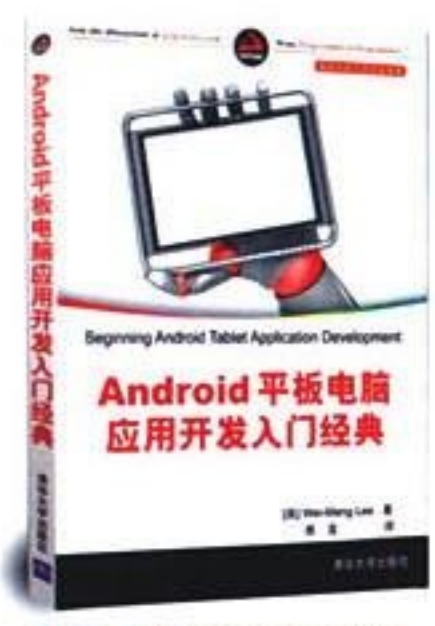
更多创赢开发图书推荐：



书号：9787302301516
定价：68.00元



书号：9787302299943
定价：69.80元



书号：9787302299967
定价：49.00元

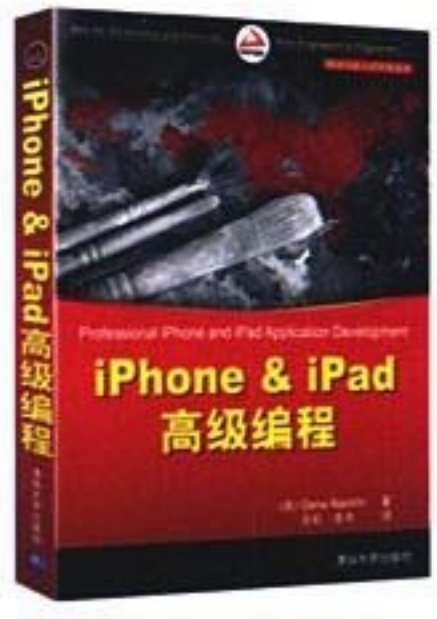
书号：9787302295440
定价：49.00元

书号：9787302286790
定价：49.80元

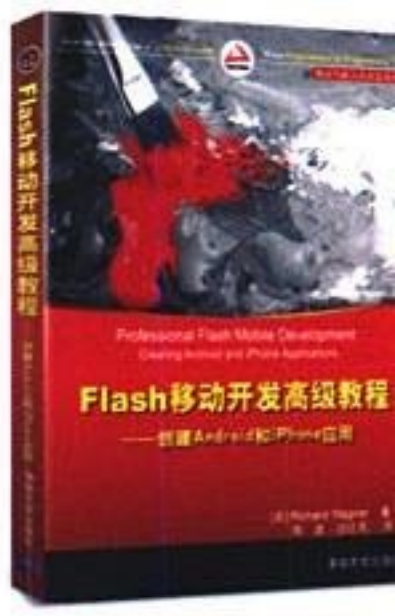
书号：9787302301059
定价：69.00元

书号：9787302303039
定价：59.80元

书号：9787302280507
定价：48.00元



书号：9787302274452
定价：59.80元



书号：9787302282099
定价：48.00元

清华大学出版社